VALIDATION OF A DATA ASSIMILATION TECHNIQUE FOR AN URBAN WIND MODEL

by

Thomas Michael Booth

A thesis submitted to the faculty of
The University of Utah
in partial fulfillment of the requirements for the degree of

Master of Science

Department of Mechanical Engineering

The University of Utah

May 2012

The University of Utah Graduate School

STATEMENT OF THESIS APPROVAL

The thesis of **Thomas Michael Booth**

has been approved by the following supervisory committee members:

Eric Pardyjak	, Chair	**10/25/2011** Date Approved
Kuan Chen	, Member	**10/25/2011** Date Approved
Meredith Metzger	, Member	**10/25/2011** Date Approved

and by **Timothy Ameel**, Chair of the Department of **Mechanical Engineering**

and by Charles A. Wight, Dean of The Graduate School.

ABSTRACT

The Quick Urban and Industrial Complex (QUIC) dispersion modeling system has been developed to calculate wind and concentration fields in cities with buildings explicitly resolved. As opposed to other models which are either limited to a simplified gaussian plume without buildings or are computationally expensive and take weeks to grid and solve. The focus of this paper is a new data assimilation technique that improves QUIC-URB, a fast response three-dimensional (3D) diagnostic urban wind model. The QUIC-URB modeling system discussed in this paper was adapted from a previous version, which initialized the flow field with horizontally uniform velocities based on wind speed and wind direction information obtained from a single measurment upwind of an urban area. Previous urban studies have shown that cities are often subject to large scale spatially varying inflows. To account for this spatial heterogeneity, a simple Quasi-3D Barnes Objective Map Analysis Scheme (a Gaussian weighted averaging technique), which initializes the flow field based on multiple sensors and soundings located around the urban area has been implemented. This wind field is then modified by QUIC-URB's empirical building flow parameterizations to model the flow around individual buildings. The final flow field is then obtained by ensuring mass conservation.

This work is a validation of this multisensor data assimilation QUIC-URB model. The analysis shows QUIC-URB solutions compared to results of a hybrid Reynolds Averaged Navier-Stokes (RANS) solution of the same urban environment using the commercial Computational Fluid Dynamic (CFD) solver, FLUENT. Nine individual vertical velocity profiles located around the urban area are extracted from the FLUENT data set to simulate soundings around three urban environments consisting of an array of containers and three different sizes

of flow altering topographies. These velocity profiles are used as input profiles for QUIC-URB's new initialization scheme. The final wind fields from QUIC-URB and FLUENT are qualitatively and quantitatively compared.

The initial implementation of this data assimilation technique captures the gross effects of nonuniform mean wind fields around urban areas well. However, there are deficiencies when ingesting localized flow. The local flow effects of buildings and other relatively small geometries are spread out beyond their applicable region when input data is sparse. Limiting these localized effects to their applicable regions is an area for future research.

This is dedicated to my father, Steven William Booth, who was an inspiration to everyone who knew him and whose wisdom and advice will last a life time.

CONTENTS

ABSTRACT . . . . . **iii**

LIST OF FIGURES . . . . . **viii**

LIST OF TABLES . . . . . **xvi**

ACKNOWLEDGMENTS . . . . . **xvii**

CHAPTERS

1. INTRODUCTION TO QUIC-URB **1**

1.1 QUIC modeling system overview 2
1.2 QUIC-URB flow solver 3

2. NONUNIFORM FLOW FIELDS **6**

2.1 Effects of spatially inhomogeneous surfaces 6
2.1.1 Advective effect 7
2.1.2 Thermal circulation systems 7
2.2 Effects of topography 8
2.2.1 Radiation loading effects 8
2.2.2 Topographically generated winds 9
2.2.3 Topographically modified winds 9

3. NEW INITIALIZATION SCHEME **11**

3.1 Barnes objective map analysis 11
3.1.1 2D Barnes interpolation scheme 12
3.2 Quasi-3D Barnes scheme 14
3.3 Boundary layer profiles 17
3.4 Limitations 20

4. VALIDATION **23**

4.1 Validation technique 24
4.2 Model geometry 25
4.3 CFD solution 27
4.3.1 Grid details 28
4.3.2 CFD validation 37
4.3.3 FLUENT inputs 43
4.3.4 FLUENT convergence 45
4.3.5 Postprocessing 49

4.4 QUIC-URB solution 53

5. RESULTS 59

5.1 Comparison with a hybrid-RANS technique 60
5.1.1 Baseline 6x7 MUST array 63
5.1.2 6x7 MUST array and the hemisphere topography 76
5.1.3 6x7 MUST array and the large wall topography 91

6. CONCLUSION 105

APPENDICES

A. DATA ASSIMILATION CODE 107

B. FLUENT INPUTS 119

C. QUIC-URB INPUT 130

D. POSTPROCESSING SCRIPTS 135

REFERENCES 158

LIST OF FIGURES

2.1 An illustration of the advective effect of air moving from one surface roughness to another surface of differing roughness. 6

2.2 An illustration of a thermal circulation system that produces sea/lake breeze. 8

2.3 An illustration of a thermal circulation effect that produces country breezes when regional winds are weak which is caused by the heat island effect. 9

2.4 Illustrations of topographically generated winds from radiation loading effects on an inclined surface during the day and the reverse cycle at night. 10

2.5 Illustrations of topographically modified winds produced by a hill and a ridge. 10

3.1 Illustration of estimated vertical velocity profiles when given single height measurements. The black arrows represent the velocity measurements at a given height and the red lines represent the estimated boundary layer profiles. 15

3.2 A 2D Barnes objective map analysis scheme is shown for the given velocity vectors in black. 16

3.3 Three 2D horizontal planar Barnes objective results stacked to illustrate the quasi-3D Barnes scheme. 17

3.4 Internal canopy velocity profiles are shown for various attenuation coefficients. 18

3.5 Streamlines are shown for a sample case with a single upwind measurement. This solution was produced by QUIC-URB without the Barnes analysis scheme. 21

3.6 Streamlines are shown for a sample case with a measurement inside a localized building wake. This solution was generated using the Barnes scheme with QUIC-URB. 22

4.1 The 6x7 container array geometry is shown. This is a small and uniformly distributed MUST array. 26

4.2 Container dimensions and spacings shown are used for the validation analysis. These dimensions are similar to the MUST container dimensions, however they have been modified to reduce the mesh resolution used for QUIC-URB. 26

4.3 Location and dimensions of the hemisphere relative to the 6x7 container array used for the validation analysis. 27

4.4 Location and dimensions of the wall relative to the 6x7 container array used for the validation analysis. 28

4.5 Oblique and plan view of the three topographies are shown for relative comparisons. 29

4.6 The mesh spacing for the triangulated surface grid of the ground is shown for the baseline geometry. The inset shows the quadrilateral surface mesh on the sides of the containers that correspond to the sides of the triangular prisms that comprises this 'cooper mesh'. 31

4.7 The triangulated surface grid of the ground and the hemisphere is shown for the geometry of the 6x7 array with a hemisphere. The vertical slice through the 'cooper mesh' shows the vertical spacing of the prisms as well as the skewness of the cells along the transition line from the ground mesh to the hemisphere mesh. 32

4.8 The triangulated surface grid of the ground and the hemisphere is shown for the geometry of the 6x7 array with a hemisphere. The vertical slice through the hybrid unstructured volume mesh shows the vertical spacing of the prisms in the 'cooper mesh' as well as the transition from the prisms to the tetrahedral cells. 34

4.9 A wider view of the vertical slice through the hybrid unstructured volume mesh shows the growth in the height of 'transition' prisms cells as a function of the size of the triangulated surface grid on the ground. The triangulated surface on the ground grows as a function of increasing horizontal distance away from the hemisphere and 6x7 array similar to that shown in Figure 4.6. 35

4.10 University of Hamburg wind tunnel test [1] of the MUST configuration. The direction of flow is coming from the guide veins (shown in the top center of the picture) over the small roughness elements and then through the scaled 10x12 MUST array. 38

4.11 The plan view of the MUST wind tunnel test model [1] is shown with two of the profile measurement locations used for this validation. The arrow shows the direction of the incoming flow. 39

4.12 Normalized vertical velocity and turbulent kinetic energy profile comparisons at the inlet for the wind tunnel test [1], MISKAM simulation [2], and the FLUENT simulation are compared. A container was also plotted to give a reference scale. $u_{ref} = 1$ m/s, $z_{ref} = 7.29$ meters 41

4.13 Normalized vertical U and W velocity profile comparisons of the wind tunnel test [1], the refined MISKAM simulation [2], and the three following types of grids used at location 3 of the wind tunnel test shown in the inset: a hybrid unstructured grid, a 'cooper grid' constructed with triangulated prisms with a 0.5 meter horizontal refinement and another 'cooper grid' with a 0.25 meter horizontal refinement. $u_{ref} = 1$ m/s, $z_{ref} = 7.29$ meters 42

4.14 Normalized vertical U and W velocity profile comparisons of the wind tunnel test [1], the refined MISKAM simulation [2], and the three following types of grids used at location nine of the wind tunnel test shown in the inset: a hybrid unstructured grid, a 'cooper grid' constructed with triangulated prisms with a 0.5 meter horizontal refinement and another 'cooper grid' with a 0.25 meter horizontal refinement. $u_{ref} = 1$ m/s, $z_{ref} = 7.29$ meters 43

4.15 The total force convergence history for ground (upper plot) and the containers (lower plot) of the baseline 6x7 array geometry starting with the steady state solver and then switching over to a time-accurate scheme at 3,000 iterations. 47

4.16 The convergence history of the scaled residual as given by equation 4.7 for the baseline geometry and showing the steady state solution being switched over to the time-accurate scheme at 3,000 iterations. . 48

4.17 The total force convergence history for ground (upper plot) and the containers (lower plot) of the 6x7 array with a wall starting with the steady state solver and then switching over to a time-accurate scheme at 4,000 iterations. 49

4.18 The convergence history of the scaled residual as given by equation 4.7 for the 6x7 array with a wall and showing the steady state solution being switched over to the time-accurate scheme at 4,000 iterations. 50

4.19 The total force convergence history for ground (upper plot) and the containers (lower plot) of the 6x7 array with a hemisphere starting with the steady state solver and then switching over to a time-accurate scheme at 4,000 iterations. 51

4.20 The convergence history of the scaled residual as given by equation 4.7 for the 6x7 array with a hemisphere and showing the steady state solution being switched over to the time-accurate scheme at 4,000 iterations. 52

4.21 Illustration of the "sensors" in and around the container array extracted from the FLUENT solution and ingested into QUIC-URB. Each one is labeled and the measurements on the side of the axes show the coordinates of each row and column of the sensors. 54

4.22 Velocity profiles extracted from the FLUENT solution of the baseline 6x7 array at each of the nine "sensor" locations, where $u_{ref} = 1$ m/s and $z_{ref} = 7.29$ meters. 56

4.23 Velocity profiles extracted from the FLUENT solution of the 6x7 array with a hemisphere at each of the nine "sensor" locations, where $u_{ref} = 1$ m/s and $z_{ref} = 7.29$ meters. 57

4.24 Velocity profiles extracted from the FLUENT solution of the 6x7 array with a wall at each of the nine "sensor" locations, where $u_{ref} = 1$ m/s and $z_{ref} = 7.29$ meters. 58

5.1 Streamlines and surface pressure contours for the baseline 6x7 array case from the FLUENT solution. 60

5.2 Streamlines and surface pressure contours for the 6x7 array and hemisphere from the FLUENT solution. 61

5.3 Streamlines and surface pressure contours for the 6x7 array and wall from the FLUENT solution. 62

5.4 Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 0.25 meters for the baseline 6x7 array. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 64

5.5 Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 1.25 meters for the baseline 6x7 array. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 65

5.6 Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 2.75 meters for the baseline 6x7 array. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 66

5.7 Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 5.25 meters for the baseline 6x7 array. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 67

5.8 Contour slice of the difference in the spanwise (U) velocity of the FLUENT and QUIC-URB solutions at a height of 1.25 meters for the baseline 6x7 array. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 68

5.9 Contour slice of the difference in the vertical (W) velocity of the FLUENT and QUIC-URB solutions at a height of 1.25 meters for the baseline 6x7 array. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 69

5.10 Contour slice of the difference in the streamwise (V) velocity of the single sensor QUIC-URB and multisensor QUIC-URB solutions at a height of 1.25 meters for the baseline 6x7 array. Each subplot shows the position of the "sensors" ingested into multisensor QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 72

5.11 Contour slice of the difference in the spanwise (U) velocity of the FLUENT and QUIC-URB solutions at a height of 0.25 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 77

5.12 Contour slice of the difference in the spanwise (U) velocity of the FLUENT and QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 78

5.13 Contour slice of the difference in the spanwise (U) velocity of the FLUENT and QUIC-URB solutions at a height of 2.75 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 79

5.14 Contour slice of the difference in the spanwise (U) velocity of the FLUENT and QUIC-URB solutions at a height of 5.25 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 80

5.15 Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 0.25 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 81

5.16 Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 82

5.17 Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 2.75 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 83

5.18 Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 5.25 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 84

5.19 Contour slice of the difference in the vertical (W) velocity of the FLUENT and QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 85

5.20 Contour slice of the difference in the spanwise (U) velocity of the single sensor QUIC-URB and multisensor QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into multisensor QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 88

5.21 Contour slice of the difference in the streamwise (V) velocity of the single sensor QUIC-URB and multisensor QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into multisensor QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 89

5.22 Contour slice of the difference in the vertical (W) velocity of the single sensor QUIC-URB and multisensor QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into multisensor QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 90

5.23 Contour slice of the difference in the spanwise (U) velocity of the FLUENT and QUIC-URB solutions at a height of 0.25 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 92

5.24 Contour slice of the difference in the spanwise (U) velocity of the FLUENT and QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 93

5.25 Contour slice of the difference in the spanwise (U) velocity of the FLUENT and QUIC-URB solutions at a height of 2.75 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 94

5.26 Contour slice of the difference in the spanwise (U) velocity of the FLUENT and QUIC-URB solutions at a height of 5.25 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 95

5.27 Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 0.25 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 96

5.28 Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 97

5.29 Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 2.75 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 98

5.30 Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 5.25 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 99

5.31 Contour slice of the difference in the spanwise (U) velocity of the single sensor QUIC-URB and multisensor QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into multisensor QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 102

5.32 Contour slice of the difference in the streamwise (V) velocity of the single sensor QUIC-URB and multisensor QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into multisensor QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 103

5.33 Contour slice of the difference in the vertical (W) velocity of the single sensor QUIC-URB and multisensor QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into multisensor QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles. 104

LIST OF TABLES

5.1 Normalized Root Mean Square Error (NRMSE) of the FLUENT and QUIC-URB solutions for the baseline 6x7 container array at four 1.5 meter vertical sections for the spanwise (U) velocity, streamwise (V) velocity, the vertical (W) velocity and the velocity magnitude (Mag). The FLUENT solution is the estimator and the QUIC-URB solution is the estimated parameter in equation 4.2 71

5.2 Normalized Root Mean Square Error (NRMSE) of the single sensor QUIC-URB and multisensor QUIC-URB solutions for the baseline 6x7 container array at four 1.5 meter vertical sections for the spanwise (U) velocity, streamwise (V) velocity, the vertical (W) velocity and the velocity magnitude (Mag). The single sensor QUIC-URB solution is the estimator and the multisensor QUIC-URB solution is the estimated parameter in equation 4.2 73

5.3 Corrected Normalized Root Mean Square Error (NRMSE) of the FLUENT and QUIC-URB solutions for the baseline 6x7 container array at four 1.5 meter vertical sections for the spanwise (U) velocity, streamwise (V) velocity, the vertical (W) velocity and the velocity magnitude (Mag). The FLUENT solution is the estimator and the QUIC-URB solution is the estimated parameter in equation 4.2 75

5.4 Corrected Normalized Root Mean Square Error (NRMSE) of the FLUENT and QUIC-URB solutions for the 6x7 container array and hemisphere at four 1.5 meter vertical sections for the spanwise (U) velocity, streamwise (V) velocity, the vertical (W) velocity and the velocity magnitude (Mag). The FLUENT solution is the estimator and the QUIC-URB solution is the estimated parameter in equation 4.2 87

5.5 Corrected Normalized Root Mean Square Error (NRMSE) of the FLUENT and QUIC-URB solutions for the 6x7 container array and wall at four 1.5 meter vertical sections for the spanwise (U) velocity, streamwise (V) velocity, the vertical (W) velocity and the velocity magnitude (Mag). The FLUENT solution is the estimator and the QUIC-URB solution is the estimated parameter in equation 4.2 100

ACKNOWLEDGMENTS

I would like to recognize my wife Dayna for her patience, Dr. Eric Pardyjak for hiring me as his research assistant, the Department of Energy for funding this research project, the Department of Defense for funding my graduate studies and giving me my first job after graduation, the EG3 branch at NASA Johnson Space Center for letting me use thousands of computational hours on their clusters, and my parents and family who supported me.

CHAPTER 1

INTRODUCTION TO QUIC-URB

For decades scientists and engineers have developed computational models to estimate the local effects of macro-scale meteorological systems [3]. Local meteorological effects can be defined as the wind field within an urban area.

On a larger scale, many computational models have been developed to calculate the flow of a fluid around any object to predict its behavior. Since air is the most abundant fluid on earth, it is most likely the most modeled fluid on earth. Aside from being essential to life, air serves countless other purposes. It provides a medium in which planes can fly, pollen can be transported from plant to plant, and provides the force to turn windmill blades to generate electricity. Wind can also transport pollutants and toxic chemicals in cities and urban areas globally, and the effects of this kind of toxic airflow is a topic of great interest.

In many cities throughout the world it is common to have a large industrial complex or a network of complexes within tens of miles of an urban area. These industrial sites can be the sources of unwanted air pollutants. The surrounding cities are best served to investigate the common path of these pollutants and there have been many experimental tests quantifying the phenomenon surrounding these problems [4, 5, 6, 1, 7, 8, 9, 10]. Computational Fluid Dynamics (CFD) is a method commonly used to solve this type of problem [11, 12, 13, 14]. Using various numerical techniques, CFD solves the the Navier-Stokes (NS) equations (the governing equations of fluid dynamics). Two popular CFD approaches deployed to solve the transport of plumes are Reynolds Averaged Navier-Stokes (RANS) [15, 2] and Large Eddy Simulation (LES) [16] with many hybrid methods that blend these two approaches. RANS is much less computationally expensive than

LES yet still has a higher computational cost than QUIC and in emergencies a quick solution is more desirable.

The total wall time to create a grid and deliver a CFD solution is on the order of days for a properly resolved computational grid of an average urban area, as opposed to a few hours with QUIC. Once the gridding is complete, QUIC-URB can deliver a solution on a city such as Oklahoma City [17] in 10 minutes and additional research is further increasing the speed of the code [18, 19]. In most cases, such as city planning where many simulations must be run, a wall time of days is not an issue. However, there are time-critical cases where a computation that takes more than a day is too long. In the event of an accidental release of toxic chemicals from an industrial complex, a train derailment that releases toxic airborne plumes, or the release of a biological or chemical weapon the code that delivers a quick and resonably accurate solution has the highest priority.

QUIC is computationally light enough to run on a common laptop, which also provides portability. It was created, from the start, to be a fast response plume dispersion modeling system. This modeling system is based on an empirically derived set of parameterizations for single and multiple building configurations that commonly occur within cities [20]. The goal of this work is to enhance this existing capability by providing data assimilation that captures large scale spatially varying flowfields.

1.1 QUIC modeling system overview

The QUIC modeling system is comprised of three different components. The preprocessing and postprocessing user interface (QUIC-GUI), the empirically derived time averaged wind field model (QUIC-URB), and the plume dispersion solver (QUIC-PLUME).

QUIC-GUI is QUIC's Graphical User Interface (GUI) suite that enables complex urban environments to be quickly and easily entered into the project data file. This is the pre and postprocessor of the QUIC modeling system. It defines the building and vegetation geometry, the grid spacing, the meteorological conditions,

the plume source and the simulation parameters. After the inputs have been defined and the solutions have been calculated, the GUI can display the results with all of the typical fluid flow visualizations.

QUIC-URB is the flow solver of the QUIC modeling system [21] which uses the building geometry, grid resolution, and meteorological inputs defined with QUIC-GUI to initialize the flow field with a single upwind meteorological input. Next it utilizes several empirical flow parametrizations [22, 23, 24, 25, 26, 27, 28, 29] based on the Röckle [20] diagnostic wind modeling strategy to estimate the flow around common building geometries within urban areas. The equation of mass conservation is then solved for the flow field for a final solution. A scheme for modeling the traffic induced turbulence [30] has also been developed when the situation warrants it.

QUIC-PLUME is the plume dispersion solver [31] that relies on the velocity vector field created by QUIC-URB to calculate the plume concentration field of the given gas. It utilizes a modified Lagrangian random-walk (stochastic) algorithm for computing the associated gas dispersion as it is convected by the time-averaged wind field.

1.2 QUIC-URB flow solver

Before this data assimilation technique was implemented, QUICURB used the velocity magnitude and direction at a single point or a single velocity profile in which to initialize the wind field. Depending on the measurement device this could simply be wind speed and wind direction from an anemometer or it could be a complete vertical velocity profile. With a single velocity vector the analyst is left to determine the velocity profile exponent (p) or a roughness height ($z_\circ$) to estimate the boundary layer vertical velocity profile using either equation 1.1 or 1.2

$$u(z) = u_{measured}\left(\frac{z}{H_{site}}\right)^{p} \tag{1.1}$$

$$u(z) = u_{measured}\left(\frac{ln\left(\frac{z}{z_o}\right)}{ln\left(\frac{H_{site}}{z_o}\right)}\right) \tag{1.2}$$

where $u_{measured}$ is the velocity measured at the height of the sensor H_{site}.

Once this vertical velocity profile is estimated the entire wind field is initialized with a horizontally uniform velocity profile. Then, QUIC-URB analyzes the building configurations and applies empirical flow parameterizations where appropriate based on this single upwind measurement. These local parameterizations use the direction and magnitude of the initial flow to determine the size and direction of local velocity near the buildings. There is ongoing research to increase the speed and accuracy of the QUIC modeling system for flow around different configurations of urban areas. [29, 12, 19]

The goal of this paper is to further validate [32] that this data assimilation technique increases the accuracy of the mean flow field before the parameterizations define the local building level flow field. This initial flow field will also be known as the preparameter flow field. The current single measurement scheme has room for improvement since, as previously stated, cities can be subject to large scale spatially varying flows [4, 5]. Also the empirical flow parameterizations will reach a level of accuracy where any further reduction in error will be negligible compared to the error introduced by the preparameter wind field for spatially varying flows. In other words, the preparameter flow field may have the velocity and direction of the wind incorrect at any given building. Since the empirical flow parameterizations rely on the preparameter flow magnitude and direction at the locations of the buildings to setup the localized flow around the building, the parameterizations will be incorrect regardless of their inherent accuracy.

This paper will layout the steps taken to increase the accuracy of the preparameter wind field initialization scheme for wind fields with spatially varying flow conditions. This new scheme will be explained in more detail in sections 3.1 and 3.2. Then several different levels of spatially varying flow fields with a simplified 6x7 block array representing an urban area similar to the Mock Urban Setting Test (MUST) [8] will be solved with a commercial CFD code (FLUENT) which will

provide the meteorological inputs for the QUIC modeling system. The FLUENT and QUIC-URB solutions will be quantitatively and qualitatively compared to provide a measure of accuracy of the data assimilation technique built into the QUIC modeling system.

CHAPTER 2

NONUNIFORM FLOW FIELDS

This chapter describes a few of the physical phenomena behind large scale spatially varying flows and the different conditions that cause these flows.

Spatial surface inhomogeneity and varying topography are two of the main topics to be addressed towards nonuniform flow fields. Both of these encompass several different physical flow phenomena. The complexity of modeling each of these separate effects quickly reaffirms the need for a better data assimilation method that captures the bulk effect of these on the mean wind field.

2.1 Effects of spatially inhomogeneous surfaces

Advective effects and thermal circulation systems are two separate sub-categories of spatial inhomogeneity. The advective effect is the horizontal movement of air from one type of surface to another as illustrated in Figure 2.1, while thermal circulation systems are defined as the circulation of air via dissimilar surface properties [33].

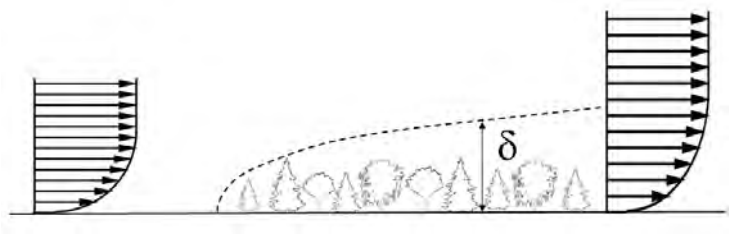

Figure 2.1. An illustration of the advective effect of air moving from one surface roughness to another surface of differing roughness.

2.1.1 Advective effect

The Clothesline Effect, Leading Edge Effect and the Oasis Effect are three main advective effects [33].

The Clothesline Effect is restricted to the flow of air through a vegetative field or forest. This usually occurs at the edge of a crop field surrounded by a warmer, drier field. This effectively dries out the soil around the border of the crop field and enhances the evaporation rate.

The Leading Edge or "Fetch" Effect occurs at the separation point between a flat arid region and a moist vegetative region. The evaporation rate at this edge rises suddenly and is quite significant, but asymptotically drops to some higher than initial constant rate. The density of air is inversely proportional to the evaporation rate.

The Oasis Effect occurs at a body of water in the middle of a dry arid region. The climate above the water is cooler due to evaporative cooling. There is a continual air-to-oasis inversion temperature gradient driving the heat flux downward.

2.1.2 Thermal circulation systems

Land and Sea/Lake breezes are caused by temperature differences between the two. The air above the water is cooler during the day for several reasons. The water turns much of the energy from the sun into latent heat rather than sensible heat, (i.e., the energy is used to evaporate the water). The water also allows transmission of short wave radiation to considerable depths which dissipates this energy into a large volume. Lastly it has a high thermal mass because of its higher heat capacity.

The air above land becomes hotter during the day than the air over the water. The air closest to the ground is heated more rapidly than the air several meters up, so the air near the ground becomes more buoyant and rises. With the air over the land rising it pulls in the cooler more dense air from the body of water creating a breeze. This phenomenon is illustrated in Figure 2.2. At night the land cools quicker than the water and the cooler more dense air over the land sinks

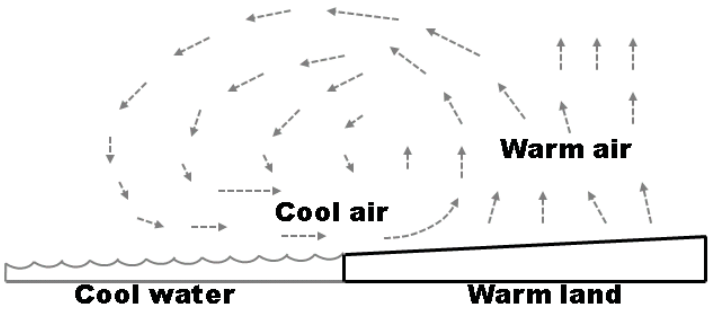

Figure 2.2. An illustration of a thermal circulation system that produces sea/lake breeze.

and is driven toward the warmer more buoyant air over the water. This creates a weaker breeze that blows towards the water at night.

Country breezes are generated by the warmer climate a city can sometimes have when regional winds are weak. A city can have a significant higher temperature due to the thermal mass of the buildings, roads, sidewalks, and parking lots compared to the same size of ground covered in vegetation. This is known as an urban heat island [34]. Just like the sea breeze the hotter, more buoyant air above the city rises while the cooler air surrounding the city must rush in to fill its place. This wind will always be directed toward the city while regional wind are weak. This effect is illustrated in Figure 2.3.

This breeze can also occur out of a strand of trees or a forest next to unshaded surroundings or fields. This is also due to the temperature differences of the separate regions.

2.2 Effects of topography

Spatially varying topography can generate or modify the climate through radiation loading effects, topographically generated winds, and topographically modified winds.

2.2.1 Radiation loading effects

Radiation from the sun heats surfaces perpendicular to its rays up to five to six times as much as surfaces at an angle [33]. The amount of energy transferred

Figure 2.3. An illustration of a thermal circulation effect that produces country breezes when regional winds are weak which is caused by the heat island effect.

to the surface and the surrounding air is a function of the surface angle and the position of the sun. This difference in surface temperatures and thus the air above the surfaces can create a breeze blowing in the direction of the warmer surface as in the case of sea breezes.

2.2.2 Topographically generated winds

Topographically generated winds are caused by the radiation loading effects on inclined surface topography. On an inclined slope or mountain adjacent to a flat valley floor, the slopes of the mountains absorb more energy during the day than the valley floor. The air is pulled from the cooler environment to the warmer slopes. As the air leaves the valley and travels up the mountains more air is drawn down into the valley. This cycle is reversed at night due to the emission of long wave radiation [33]. Illustrations of these topographically generated winds is shown in Figure 2.4.

2.2.3 Topographically modified winds

Local topography can obviously modify existing winds. A hill or a ridge in the middle of a flat field, a series of mountains, a set of gully's or depressions all directly affect the surrounding flow field by causing the wind to flow around them.

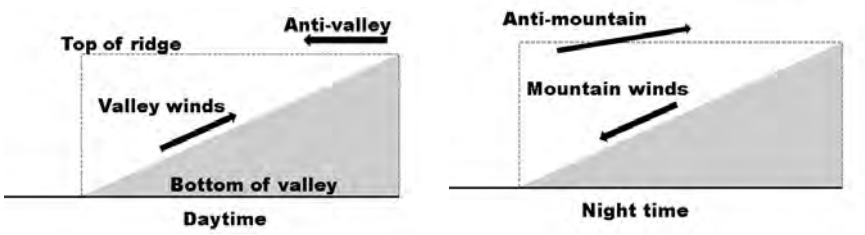

Figure 2.4. Illustrations of topographically generated winds from radiation loading effects on an inclined surface during the day and the reverse cycle at night.

Figure 2.5 shows illustrations of a hill and a ridge topography that modify the surrounding wind.

Wind flowing from smooth to rough terrain will also undergo a slight change in direction if the wind is not perpendicular to the boundary line between the different surfaces. The winds direction will shift toward this transition line when flowing from the smooth surface to the rough surface and away from the line when flowing from the rough to the smooth.

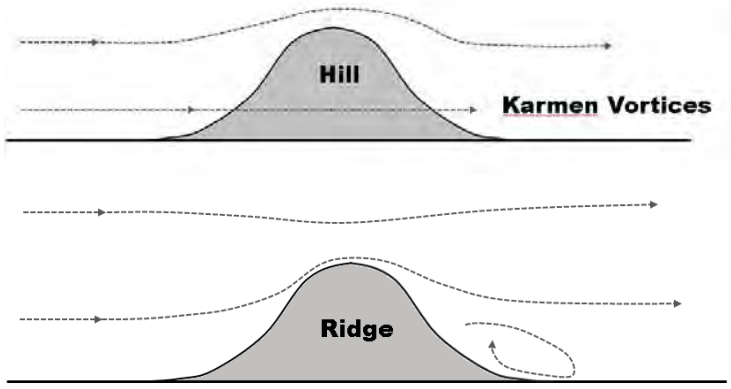

Figure 2.5. Illustrations of topographically modified winds produced by a hill and a ridge.

CHAPTER 3

NEW INITIALIZATION SCHEME

This chapter describes an interpolation technique that is used for large scale satellite meteorlogical data assimilation and how it is modified to work with wall bounded flows for the QUIC modeling system. The purpose of this is to estimate a temporally and spatially averaged flow field from meteorological measurements around cities with the goal of capturing the significant flow spatial inhomogeneities described in the previous chapter. Several boundary layer velocity profiles are also reviewed here to estimate a full vertical velocity profile at sites measuring only the velocity and direction of the wind.

3.1 Barnes objective map analysis

A data assimilation technique for meso-scale analysis of high quality fields of satellite meteorological measurements is presented here for the purpose of capturing previously discussed inhomogeneities in the flow field of urban areas. The technique investigated here is based on the interactive Barnes objective map analysis scheme [35] for use with satellite and conventional data which was based on the original Barnes objective analysis technique [36]. This method was chosen over other data assimilation techniques for its ease of implementation into QUIC-URB since it had previously been implemented in two dimensions for QUIC-URB.

In the case of satellite cloud motion wind data, high quality data sets are available for mesoscale analysis. The problem of accurately assigning cloud heights and placing data onto a coordinate grid prior to their insertion into numerical prediction models was solved by the Barnes objective map analysis scheme.

This interpolation scheme works well for mesoscale atmospheric flow, however the sparseness of available meteorological data and the proximity of wall bounded flows discourages the use of the Barnes scheme in a full 3D scope. The Barnes scheme does not have a viscous wall boundary condition built into it, so without measurements at the ground to bound the data interpolation technique the flow field will not obey the "no-slip" boundary condition. In addition to this, the gaussian weighted averaging technique would not calculate a correct viscous boundary layer even if the wall velocities were set to zero. To prevent nonzero velocities at the ground and to better estimate the viscous boundary layer velocities, the Barnes scheme is modified. This modification limits the Barnes scheme to a two-dimensional (2D) scheme which only interpolates data in planes parallel to the ground plane. This new scheme will be known as the 2D Barnes scheme in this paper.

To create the no-slip boundary condition and the viscous boundary layer the 2D Barnes scheme is modified to utilize common boundary layer velocity profile estimations. Instead of the velocity being interpolated in three dimensions this modified 2D scheme will use the velocity given by typical boundary layer velocity profiles at each plane parallel to the ground. This process will be explained further in the following sections and it will be called the quasi-3D Barnes scheme in this paper.

3.1.1 2D Barnes interpolation scheme

The Barnes scheme is explained in more detail in Koch [35]; however a summary will be given here. The scheme produces a Gaussian weighted average of the form:

$$w_m = exp\left(\frac{-r_m^2}{\kappa}\right)^2 \tag{3.1}$$

where r_m is the distance between the computational grid point and the location of the data point, and κ is the parameter which determines the shape of the filter response function. The computational grid discussed here is the QUIC-URB grid.

This analysis scheme steps through two different iterations while interpolating the measured data to the computational grid. During the first iteration the scheme calculates the computational data spacing. The computed data spacing Δ_n is calculated by the average distance between each data point and its nearest neighbor. The weight parameter $k_\circ$ is calculated from Δ_n by equation 3.2 given below.

$$\kappa_\circ = 5.052 \left(\frac{2\Delta_n}{\pi} \right)^2 \tag{3.2}$$

The first computational iteration calculates the weighted average of the data at each grid cell location using $\kappa_\circ$ in equation 3.1 which produces an initial velocity field $u_\circ(x, y)$ from equation 3.3

$$u_\circ(i, j) = \frac{\sum_{n=1}^{M} w_m u_{sensor}(x, y)}{\sum_{n=1}^{M} w_m}, \tag{3.3}$$

where $u_{sensor}(x, y)$ is the velocity at each sensor location, and M is the number of sensor data points located within the computational domain. A new velocity $u_{int}(x, y)$ at each sensor location is linearly interpolated from the four surrounding grid points in the initial interpolated velocity field $u_\circ(i, j)$.

Barnes [36] increased the computational efficiency of this analysis scheme by implementing a single 'correction' iteration upon the initial interpolated field $u_\circ(i, j)$, rather than making several more iterations. The correction iteration is accomplished by decreasing the value of κ to a new 'correction' pass value of

$$\kappa_1 = \gamma \kappa_\circ. \tag{3.4}$$

where γ $(0.2 \leq \gamma \leq 1)$ is the numerical convergence parameter that enables a high degree of convergence between the initial measured values ($u_{sensor}(x, y)$ and the second iteration interpolated field $u_{final}(i, j)$). The convergence parameter is an adjustment to determine the amount of acceptable error between the actual sensor readings and the final interpolated wind field at the location of the sensors. For instance the minimum error between the initial measurements $u_{sensor}(x, y)$

and the final interpolated velocity field $u_{final}(i,j)$ at each sensor location is given by γ=0.2, while the maximum error is given by γ=1. A γ of 0.2 was used for this current model. More details about the behavior of γ are given in [35]. This parameter should be the focus of future research.

The final velocity field $u_{final}(i,j)$ is calculated using the difference between the sensor data $u_{sensor}(x,y)$ and the new interpolated values $u_{int}(x,y)$ located at the sensor locations in the following equation:

$$u_{final}(i,j) = \frac{\sum_{n=1}^{M} w_m \left(u_{sensor}(x,y) - u_{int}(x,y)\right)}{\sum_{n=1}^{M} w_m} \tag{3.5}$$

3.2 Quasi-3D Barnes scheme

As previously mentioned the interactive Barnes objective analysis scheme was not well suited for wall bounded flows with sparse data. The previous section described the 2D restriction necessary to let this Barnes scheme satisfy the no-slip boundary condition at the ground. It is necessary for this scheme to only interpolate data at planes parallel with the ground, however this comes with its drawbacks. For instance, most meteorological measurements around cities do not lie in the same horizontal plane, therefore without the ability to interpolate in the vertical direction this scheme is not effective. However, it is possible to estimate a vertical velocity profile at the location of the measurement using the velocity and height of the measurement location. The applicable boundary layer velocity profiles are discussed later in section 3.3.

Since this scheme has been limited from interpolating in the Z-direction it can not be considered a 3D interpolation scheme. However, with the use of some commonly known boundary layer profiles as an estimate of the vertical velocity gradient this scheme will have an estimate of the Z-directional velocity gradient, therefore it will be known as the quasi-3D Barnes scheme in this paper.

The quasi-3D Barnes map analysis scheme begins by approximating a vertical velocity profile at each location supplied with meteorological data. The vertical

velocity profiles are approximated by either a logarithmic or an exponential profile as previously given in equations 1.1 and 1.2 for typical flat plate boundary layers.

Figure 3.1 shows a graphical example of importing data for three different sensors and approximating their vertical velocity profiles. The black arrows represent the wind speed and direction at each sensor location and the red lines represent the approximated profiles.

To estimate a 3D flow field a 2D horizontally planar flow field as shown in Figure 3.2 is calculated at every grid cell height up to the full height of the computational domain. The black arrows in this figure labeled S1, S2, and S3 represent the velocity at the given cell height from the boundary layer profile

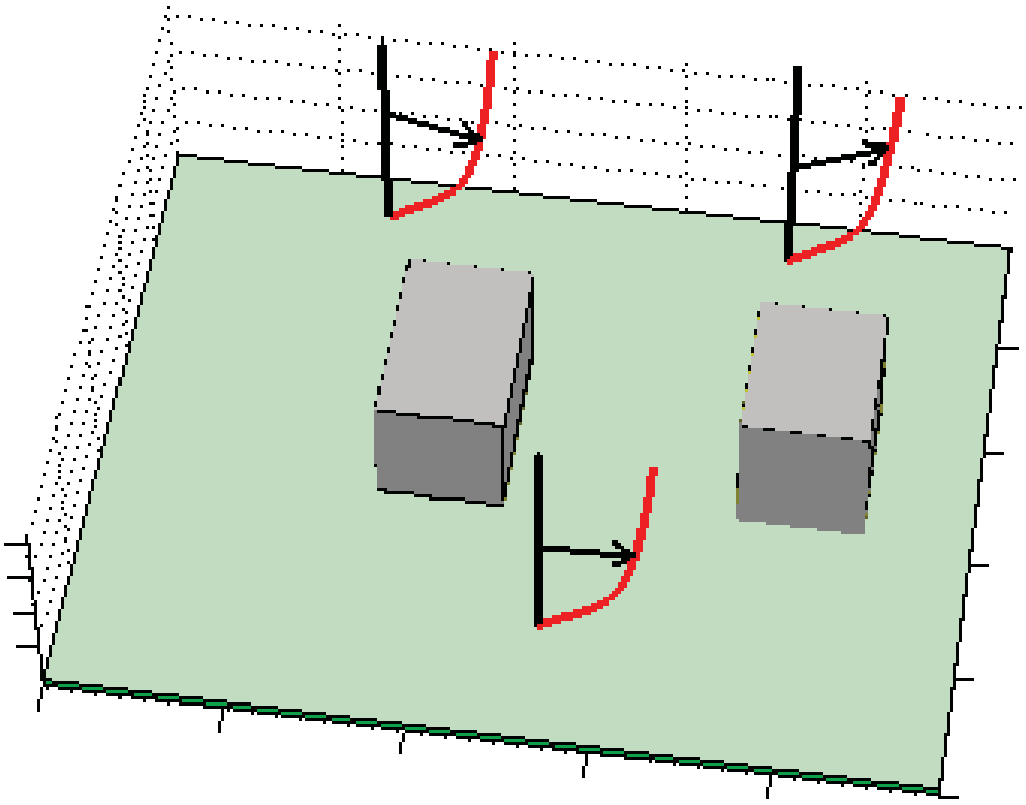

Figure 3.1. Illustration of estimated vertical velocity profiles when given single height measurements. The black arrows represent the velocity measurements at a given height and the red lines represent the estimated boundary layer profiles.

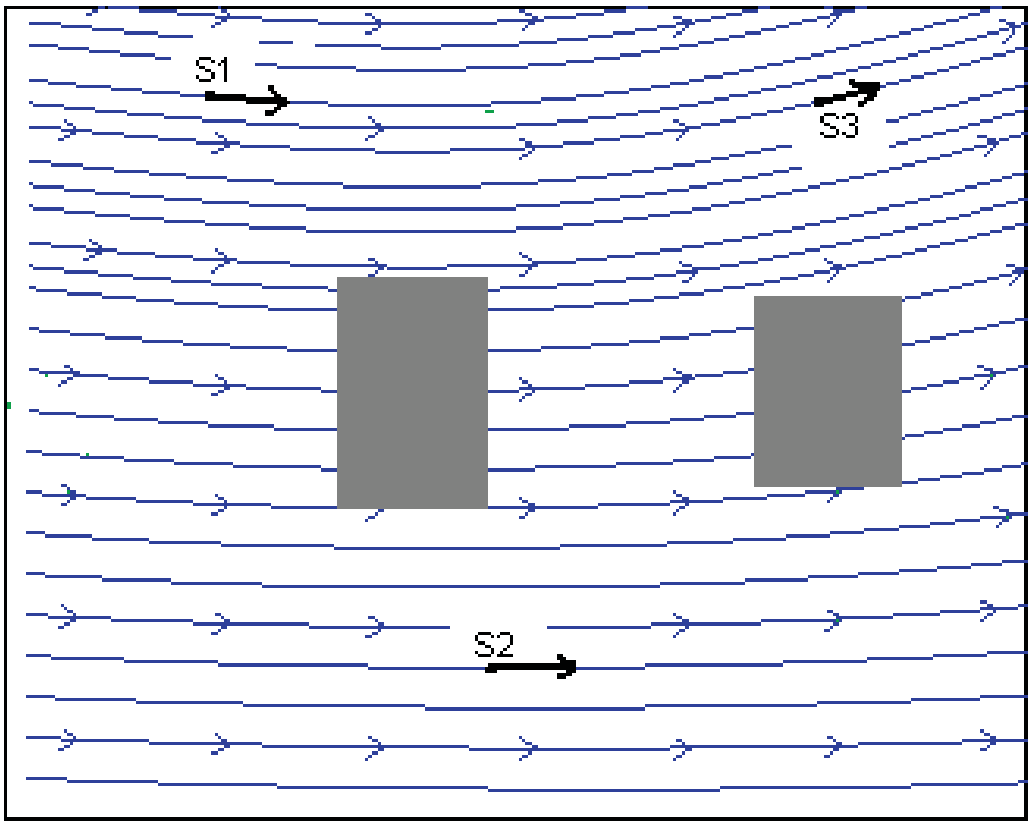

Figure 3.2. A 2D Barnes objective map analysis scheme is shown for the given velocity vectors in black.

estimations from the different sensors.

The 2D Barnes objective map analysis scheme calculates the 2D horizontally planar flow field at each grid cell height using the approximated values previously calculated for the vertical velocity profiles. The resulting flow field is comprised of these 2D horizontally planar flow fields "stacked-up" on top of each other as shown in Figure 3.3. The stacked horizontally planar flow fields still observe the "no-slip" condition of the wall and include a viscous boundary layer.

The FORTRAN source for this quasi-3D Barnes scheme is located in Appendix A.1.

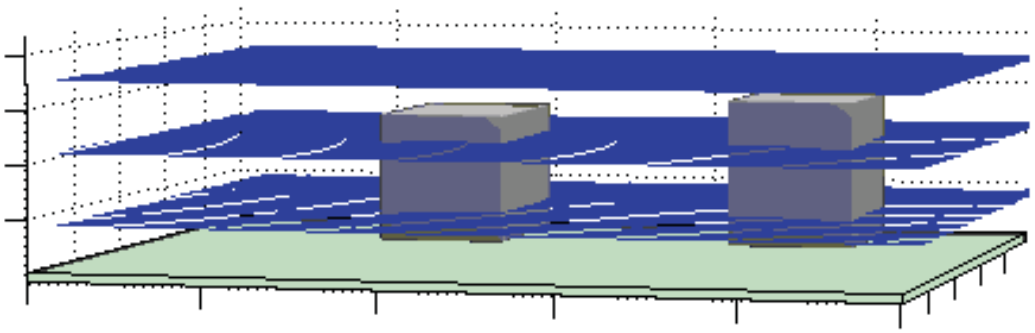

Figure 3.3. Three 2D horizontal planar Barnes objective results stacked to illustrate the quasi-3D Barnes scheme.

3.3 Boundary layer profiles

This section discusses the boundary layer profiles that can be used in the process of converting the 2D Barnes schemes into the quasi-3D Barnes scheme. The logarithmic boundary layer from equation 1.2 and the exponential boundary layer equation 1.1 are used for relatively flat areas with a roughness height $z_\circ$ effectively displacing the profile in the vertical direction. For measurements located inside the city, another set of equations are needed. Several papers [37, 38, 39, 40] suggest that the urban environment can be analogous to a plant canopy. When considering an area averaged velocity inside this urban canopy it is similar to that of a plant canopy. A mathematical model for the spatially averaged air flow in a vegetative canopy was developed [37]. The aerodynamic roughness of vegetation was expressed in terms of its height, density and drag characteristics. The averaged wind velocity profile inside a plant canopy or a canopy of equally spaced elements produces an exponential function height of the following form:

$$u(z) = u_H e^{a\left(\frac{z}{H}-1\right)} \tag{3.6}$$

where u_H is the velocity at the height (H) of the canopy and the attenuation coefficient a is a constant. For simple canopies, like the ones studied in this paper, the attenuation coefficient [38] is

$$a = \frac{ln\frac{V_z}{V_H}}{\frac{z}{H} - 1}. \tag{3.7}$$

The coefficient, a, is an index of the airflow response to the canopy roughness element where V_z is the wind velocity inside the canopy at height z and V_H is the velocity at the top of the canopy-free air interface, H. This relationship is valid from approximately 0.1H to H. The attenuation coefficient has also been empirically derived [39] for several different species of plant canopies. Examples of this internal canopy profile for attenuation coefficients from 0.4 to 2.0 are shown in Figure 3.4.

Additional work was performed [40] that focused on the spatially and temporally averaged velocity profiles in and above simplified urban canopies. The work also focused on the transitional regime between the canopy and the displaced logarithmic profile above it and illustrated the need for a blending function that

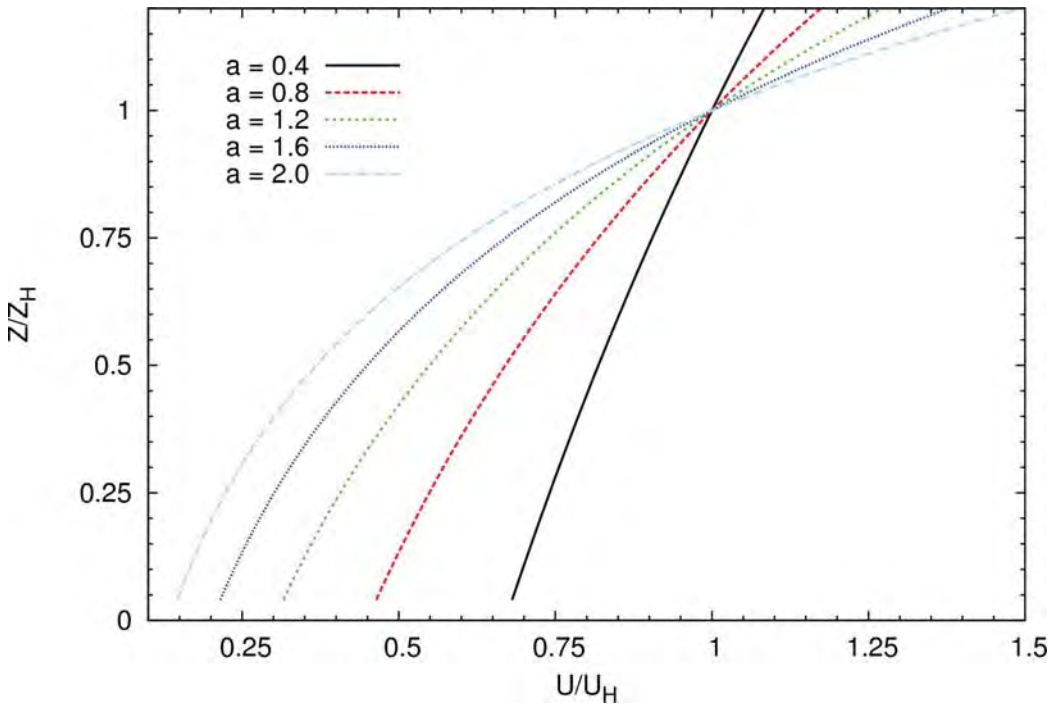

Figure 3.4. Internal canopy velocity profiles are shown for various attenuation coefficients.

smoothly transitioned between these two profiles. This work used the internal plant canopy profile given in equation 3.6 inside the canopy and the displaced logarithmic profile given in equation 3.8 as the profile above the canopy.

$$u(z) = \frac{u_*}{k} ln\left(\frac{z-d}{z_o}\right) \tag{3.8}$$

where u_* is the friction velocity, which is calculated from the wall shear stress (τ_w) and the air densitiy (ρ) shown in equation 3.9. k is the von Karman constant, d is the displacement height and z_o is the surface roughness parameter.

$$u_* = \sqrt{\frac{\tau_w}{\rho}} \tag{3.9}$$

A blending function was developed [40] to avoid a discontinuity at the junction of these two profiles somewhere near the top of the urban canopy. The blending function shown in equation 3.10 is valid from the top of the canopy (H) to a previously unknown wake diffusion height (z_w).

$$u(z) = \frac{u_*}{B} ln\left(\frac{A+Bz}{A+BH}\right) + u_H \tag{3.10}$$

where A is equal to

$$u(z) = l_c - \left(\frac{H}{z_w - H}\right)(k(z_w - d) - l_c) \tag{3.11}$$

and B is equal to

$$u(z) = l_c - \left(\frac{1}{z_w - H}\right)(k(z_w - d) - l_c) \tag{3.12}$$

and l_c is the mixing length scale at H.

This blending function is a nonlinear equation and requires an iterative solver to solve for the unknown wake diffusion height (z_w) which is somewhere above the height of the canopy (H). This entire profile requires three equations, the following known values: H, u_H, u_*, a, l_c, d, z_o and k, and requires a numerical iteration technique to solve the unknown value z_w. The bisection method is used in the QUIC-URB code to solve for z_w and is located in Appendix A.2.

3.4 Limitations

There are limitations to this Barnes scheme that come from weighting each individual measurement site the same. For equally spaced measurement sites this would give all measurement sites an equal radius of influence. The radius of influence is the distance away from the measurement site that is mostly influenced by that site. Unfortunately, this is not a correct assumption for flows with local disturbances like buildings. If the measurement sites are sparse the radius of influence of each measurement site is large. If there is building in this flow that is small compared to this radius of influence and one of the measurement sites is in the wake of the building then the wake measurement will be used beyond the wake and out to the radius of influence. This essentially diffuses the wake beyond the applicable range of the wake.

To illustrate this limitation a sample case with a single square building of height and width of 10 and 6 meters, respectively, was setup for the QUIC-URB code to solve. The first case utilized a single upwind measurement illustrated by the red arrow in Figure 3.5. The blue lines show the streamlines of the QUIC-URB solution at half the height of the building.

From this figure it is easy to see the upwind and downwind recirculation zones created by the building to the left and right of the building. These recirculation zones were calculated by the QUIC-URB empirical parameterizations. These areas have localized flow that should not extend beyond the wake of the building. Figure 3.6 illustrates the weakness of this data assimilation scheme. A measurement within the downwind wake of the building was extracted from the first velocity field and used as an input along with the previous upwind measurement with the new quasi-3D Barnes initialization scheme. The red arrows in the figure show the location, magnitude and direction of the measurements used as input.

It is apparent that this measurement taken in the wake behind a building had an effective radius of influence that was larger than its applicable localized flow region. This adverse affect can happen if the user is not careful when inputting data into this current data assimilation routine. It should be noted

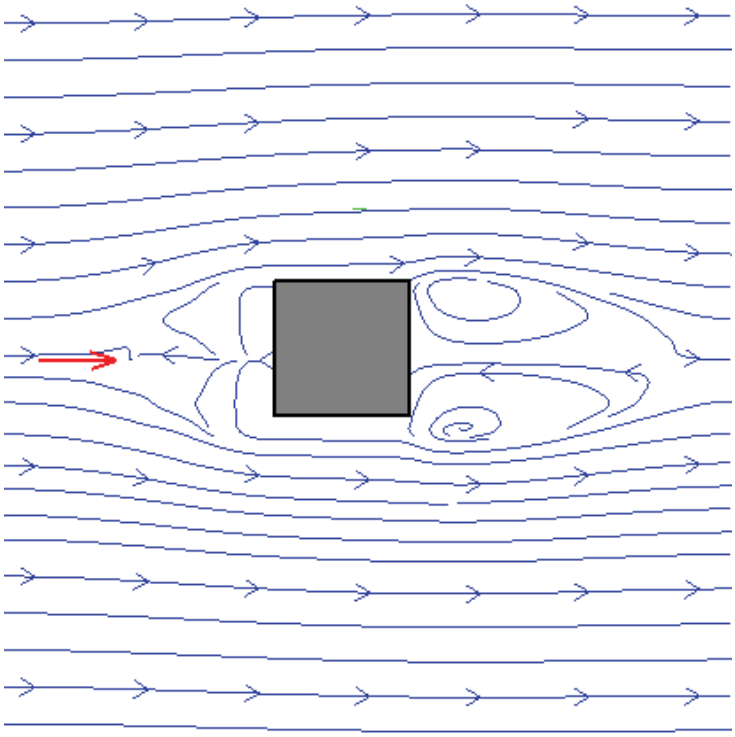

Figure 3.5. Streamlines are shown for a sample case with a single upwind measurement. This solution was produced by QUIC-URB without the Barnes analysis scheme.

that a meteorological measurement site would probably not be placed in the wake of a building or in an area that is greatly affected by localized flow. However, the user should still be aware of this limitation.

This problem is also applicable when measurement sites are within an urban canopy. In this case it would be best for the analyst using this assimilation technique to utilize the urban boundary layer profile discussed earlier.

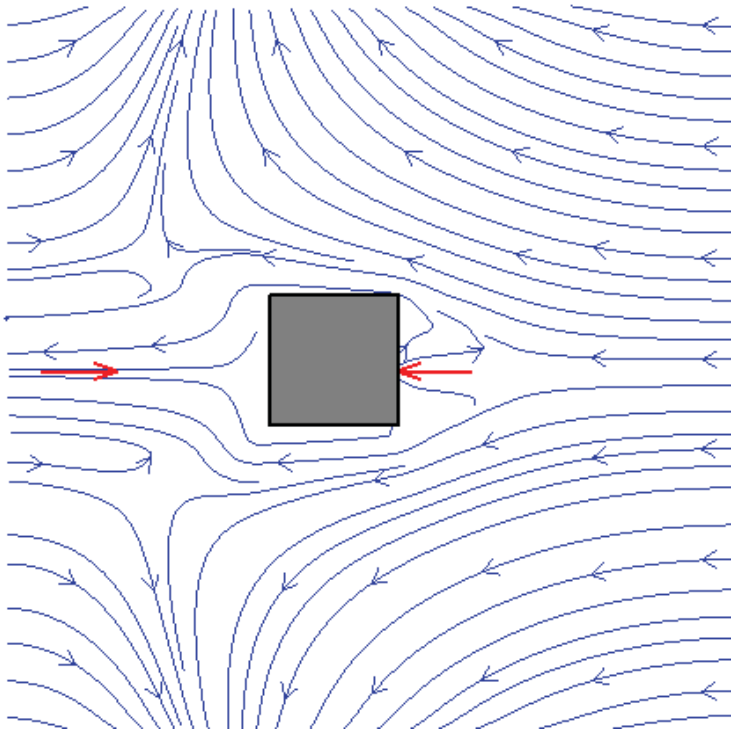

Figure 3.6. Streamlines are shown for a sample case with a measurement inside a localized building wake. This solution was generated using the Barnes scheme with QUIC-URB.

CHAPTER 4

VALIDATION

The focus of this chapter is to describe the methodology of the validation of the data assimalation technique written for QUIC-URB. The solution from this improved QUIC-URB simulation will be compared to the solution obtained from ANSYS FLUENT (a commercial Navier-Stokes CFD solver). This data assimilation validation will use a similar geometry as the Mock Urban Setting Test (MUST) [8]. MUST was performed at the U.S. Army Dugway Proving Ground (DPG) Horizontal Grid test site in the western Utah desert. Its purpose was to measure meteorological and dispersion data for a near full-scale urban environment to aid in the development and validation of plume dispersion modeling systems. The MUST was a uniform array of conex shipping containers layed out in rectangular 10x12 grid.

A 6x7 array of shipping containers was modeled in this validation to reduce computational time. This solution will provide a full velocity vector field that will be available for quantitative and qualitative comparisons with the QUIC-URB solution. It will also allow a one to one comparison of velocity vector slices through the domain in order to better understand the areas of deficiencies with the model. This 6x7 array was modeled with three terrain features to provide some insight into the advantages of the new data assimilation technique.

The Reynolds Averaged Navier-Stokes (RANS) equations given in einstein notation in equation 4.1 are the fundamental equations used by the FLUENT solver to compute the time averaged flow field.

$$\rho \frac{\partial \bar{u}_j \bar{u}_i}{\partial x_j} = \rho \bar{f}_i + \frac{\partial}{\partial x_j} \left[-\bar{p}\delta_{ij} + \mu \left(\frac{\partial \bar{u}_i}{\partial x_j} + \frac{\partial \bar{u}_j}{\partial x_i} \right) - \rho \overline{u_i' u_j'} \right] \tag{4.1}$$

The left hand side of the equation is the convection term, $\rho\bar{f}_i$ is the mean body force, $-\bar{p}\delta_{ij}$ is the mean pressure term, $\mu\left(\frac{\partial\bar{u}_i}{\partial x_j}\right)$ is the viscous stress, and $\rho\overline{u_i'u_j'}$ is the Reynolds stress that is a product from the resulting fluctuating velocity field when the RANS equations were derived from the Navier-Stokes equations using Reynolds Decomposition.

The Shear-Stress Transport (SST) Turbulence closure model [41] is used in this analysis to solve the Reynolds stress terms in equation 4.1.

The model geometry and the grid used to discretize the domain is discussed later in this chapter.

4.1 Validation technique

The validation technique presented here relies on the unsteady solution calculated from FLUENT. It has been stated that RANS is not theoretically suitable for atmospheric flows in some cases [42]. The inherent drawback of the RANS approach for atmospheric flows is the large amount of numerical damping (mainly from the turbulence models) of the naturally unsteady flow. However, for the purposes of this paper an LES solution is too computationally costly; therefore, the SST Scale Adaptive Simulation (SAS) [43] (a hybrid RANS-LES turbulence model) model is used.

A complete velocity vector field is the major advantage of using a CFD solution instead of the sparse data that experimental tests provide. However, CFD has many areas that introduce error. The aforementioned turbulence model, improper mesh refinement, poor mesh quality, improper boundary conditions, and poorly resolved time steps are all areas that can introduce uncertainty or error to a CFD solution. For these reasons it is always best to validate the CFD method to gain a reasonable confidence in the solution. A validation of this CFD method is reviewed in section 4.3.2.

The advantage of having a complete velocity vector field is the ability to extract vertical velocity profiles from any location around the container array. These extracted profiles can then be ingested into the new version of QUIC-URB and the resulting solution can be compared to the FLUENT solution at every

point in the domain. These two flow solutions can be subtracted from each other (FLUENT - QUIC-URB) at each point in the domain to analyze the differences between the two codes. The difference in the U, V, and W components the two solutions can be plotted up to highlight large differences between the codes. For a quantitative assessment of the error the Normalized Root Mean Square Error (NRMSE) is calculated for each of the three velocity components and the velocity magnitude.

$$NRMSE = \frac{\sqrt{\frac{\sum_{i=1}^{n}(x_{1,i}-x_{2,i})}{n}}}{x_{max} - x_{min}} \tag{4.2}$$

n equals the number of grid cells in the domain, $x_{1,i}$ is the estimator (FLUENT solution) and $x_{2,i}$ is the estimated parameter (QUIC-URB solution). The x_{max} and x_{min} are the maximum and minimum of the estimator.

Nine combinations of velocity profiles were fed into the data assimilation technique for QUIC-URB for each of the different terrain geometries. These will be discussed later in more detail, but in summary the measurement locations were broken up into two main catagories: internal urban canopy profiles and external canopy profiles. The external flow locations included profiles upwind, to the side of the urban array and a profile that was in the downwind wake of the array. The internal canopy profiles where near the leading edge and trailing edge of the array in the wake of building and also in the street canyons between buildings.

4.2 Model geometry

As previously stated, the MUST array geometry modeled for this research is a scaled down version of the original 10x12 container array used in the MUST experiment. The current model consists of an array that is seven containers wide and six containers deep as shown in Figure 4.1. The direction of flow is in the positive y-direction as shown by the arrow. The dimensions of each of the containers and the spacing between them in the array is illustrated in Figure 4.2.

The three terrain features simulated in this paper approximate three different scales of local topography. The 6x7 MUST array geometry on a flat plate is

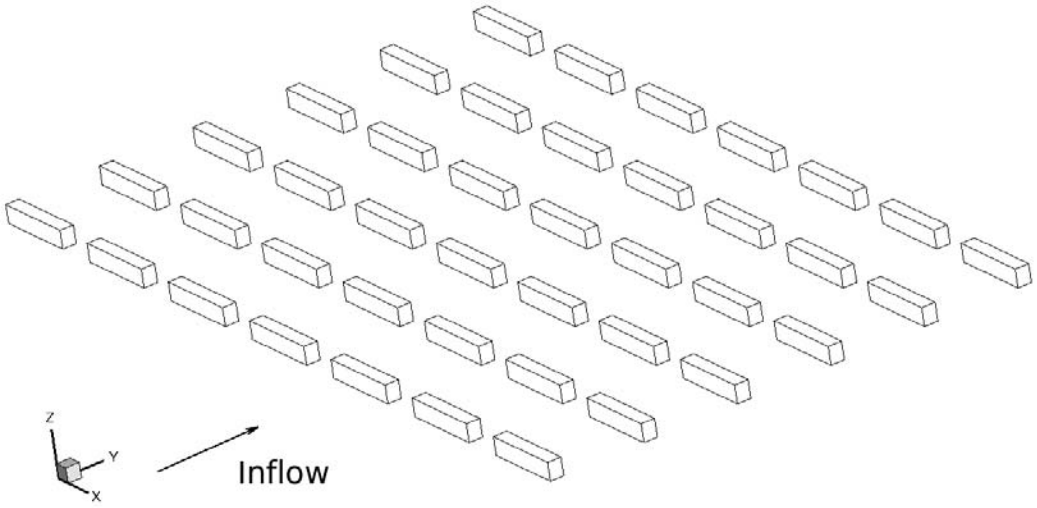

Figure 4.1. The 6x7 container array geometry is shown. This is a small and uniformly distributed MUST array.

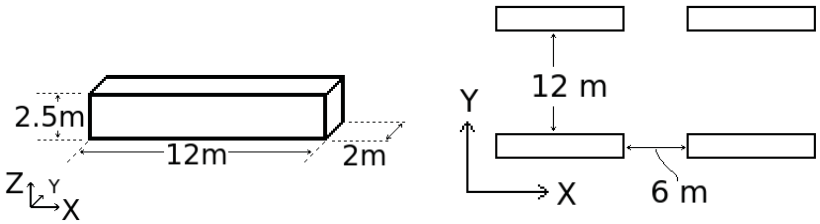

Figure 4.2. Container dimensions and spacings shown are used for the validation analysis. These dimensions are similar to the MUST container dimensions, however they have been modified to reduce the mesh resolution used for QUIC-URB.

the baseline case and represents topography that is much smaller than the city domain and is therefore negligible. The second case models a hemisphere roughly the same diameter of the city to represent local topography that is on the same order of magnitude as the city being analyzed. The third case has a large wall relatively close to the city and represents topography that is a couple of orders of magnitude larger than the city. All three of the cases have the same 6x7 container array setup, the same coordinate systems, and use the same input conditions.

Figures 4.3, 4.4 and 4.5, show the geometries, dimensions and locations of the different topographies. These geometries create an example of topographically modified winds which is only one method of many different methods discussed earlier in Chapter 2 that can cause spatially varying winds throughout the city being modeled. These other methods should be additional areas of research.

4.3 CFD solution

This section details the procedures followed to generate the CFD grid with acceptable resolution and quality, validate the CFD solution with experimental

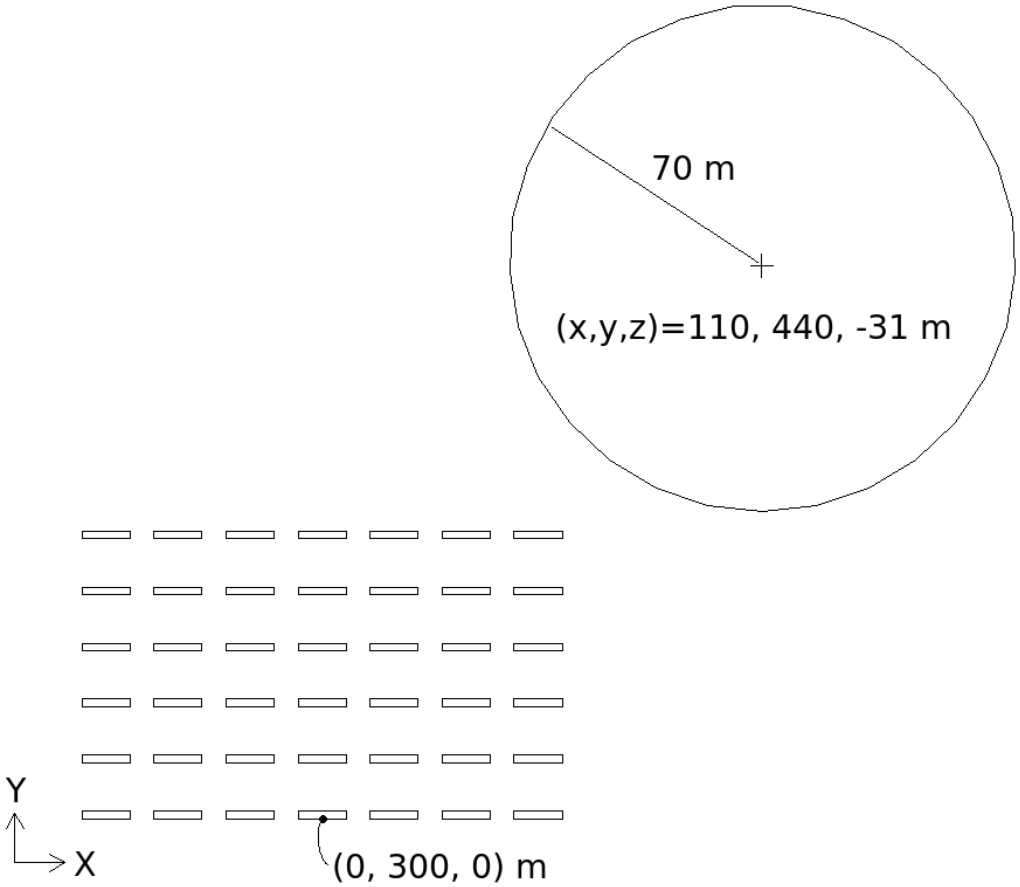

Figure 4.3. Location and dimensions of the hemisphere relative to the 6x7 container array used for the validation analysis.

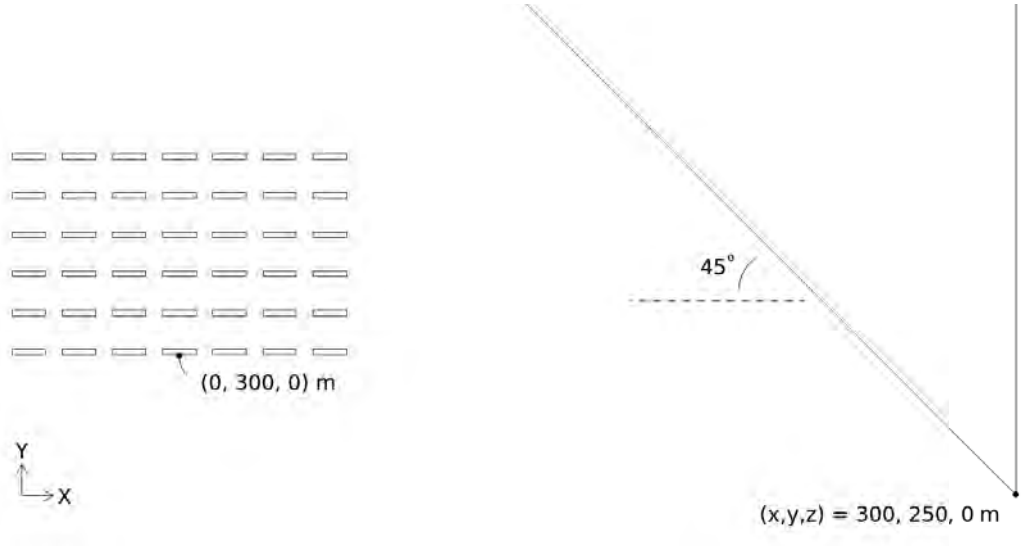

Figure 4.4. Location and dimensions of the wall relative to the 6x7 container array used for the validation analysis.

data, define a set of appropriate inputs, check the solutions for proper convergence, and postprocess the CFD solutions.

4.3.1 Grid details

Several papers [16, 2] have done grid refinement studies for CFD simulations of the MUST array. They showed that a 0.5 meter spacing near the buildings in the urban canopy was sufficiently grid converged for FEFLO-URBAN so this was the starting point for the CFD validation performed in this paper with FLUENT.

GAMBIT is the grid generation package used to create the computational meshes for all analyses in this paper. There were several different grid methodologies used to grid these three geometries. In the end there was not one consistent method that worked well for all three cases. However, they were all consistent enough through the container array and the surround area that the grids should not be a large factor in comparison between the three solutions.

The mesh technique used for the baseline and wall case is referred to as a "cooper" mesh. This method uses an unstructured triangulated surface mesh on

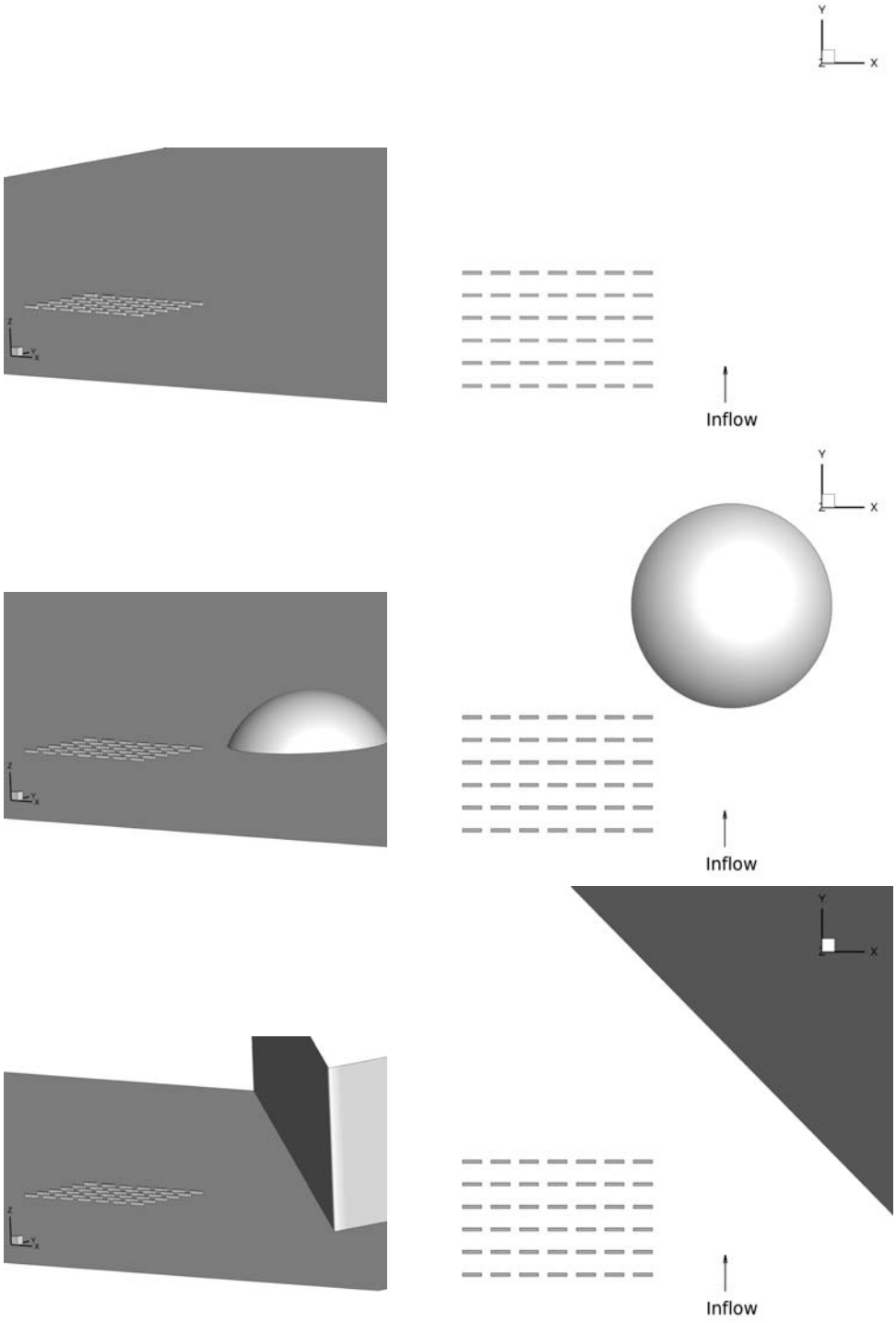

Figure 4.5. Oblique and plan view of the three topographies are shown for relative comparisons.

the ground plane however, rather than creating tetrahedral cells throughout the domain, this surface mesh is 'extruded' upward to create triangular prisms. The inset in Figure 4.6 shows the quadrilateral mesh on the sides of the containers indicating that the volume has been meshed with prisms. The cooper mesh is orthogonal to the ground, which is a desirable quality in properly capturing boundary layers and still enables cell clustering around the geometry.

Figure 4.6 shows how the sides of the domain were angled slightly to ensure that there was always flow leaving these faces. This increases solution stability since the pressure-outlet boundary conditions on these faces do not behave well with backflow on a pressure-outlet face.

Grid quantities called "size functions" were defined in order to resolve the triangulated surface mesh along the ground in the appropriate places. Several different size-functions were created to control the growth of the unstructured surface grid and refine the grid in areas of expected high gradients such as the area around the container array.

An area downwind of the array was defined to better resolve the wake coming from the containers. This refined wake area is commonly referred to as a wakebox and can be seen in Figure 4.6 as the refined black oval mesh region downwind of the array. Only one angle of incident was run on this mesh so it was not necessary to have a wakebox any bigger than this. The wakebox size function set an initial grid size of 0.25 meters on the container walls, and set a growth ratio of 1.1 to let the mesh size grow up to a maximum size of 2.0 meters within the container array and wakebox. The area outside of the wakebox (known as the farbox) has a size function that allowed the mesh to also grow at a growth ratio of 1.1 from the edges of the wakebox up to a maximum cell size of 30 meters.

The positive z-axis is defined as the cooper direction to create the cooper volume mesh from the previously described surface mesh of the ground. The vertical cell spacing for the volume cooper mesh is as follows: from the ground up to 5 meters (2 times the container height) there is a constant 0.25 meter resolution, from 5 meters to the top of the domain (125 meters) the spacing starts at 0.25

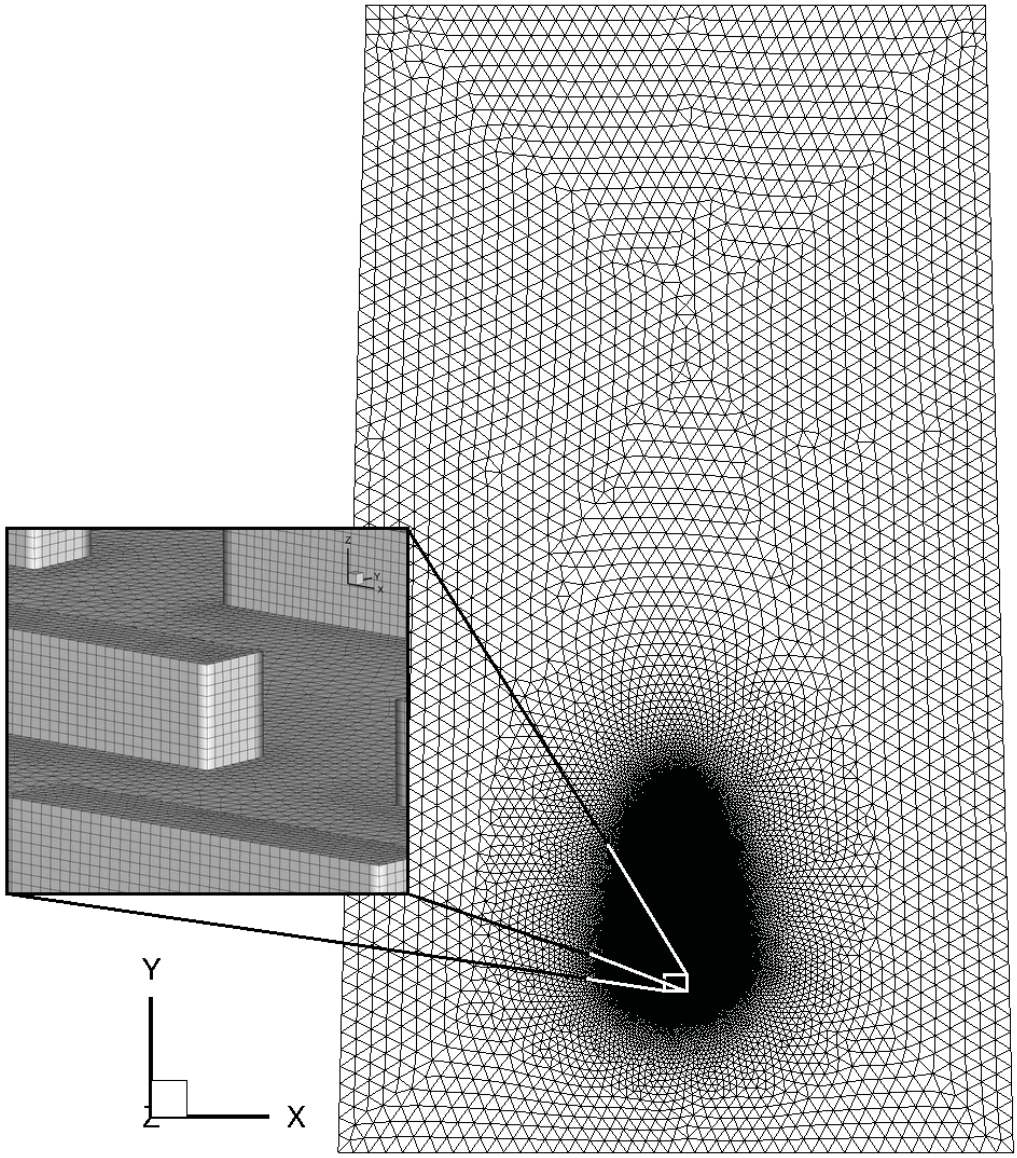

Figure 4.6. The mesh spacing for the triangulated surface grid of the ground is shown for the baseline geometry. The inset shows the quadrilateral surface mesh on the sides of the containers that correspond to the sides of the triangular prisms that comprises this 'cooper mesh'.

meters and has 41 elements which gives a stretching ratio of roughly 1.1.

This method is ideal for the baseline 6x7 container array and the wall topography since these geometries are well suited for orthogonal meshes. The baseline 6x7 array had 13.1 million cells and the array with the wall had 11.5 million cells.

The hemisphere case was also meshed with this cooper technique, unfortunately there were some highly skewed cells around the hemisphere that affected proper convergence of the solution. Figure 4.7 shows a slice through the volume of this mesh.

The highly skewed and high aspect ratio cells at the edge of the hemisphere and upward are believed to be the main reason for the convergence issues experienced.

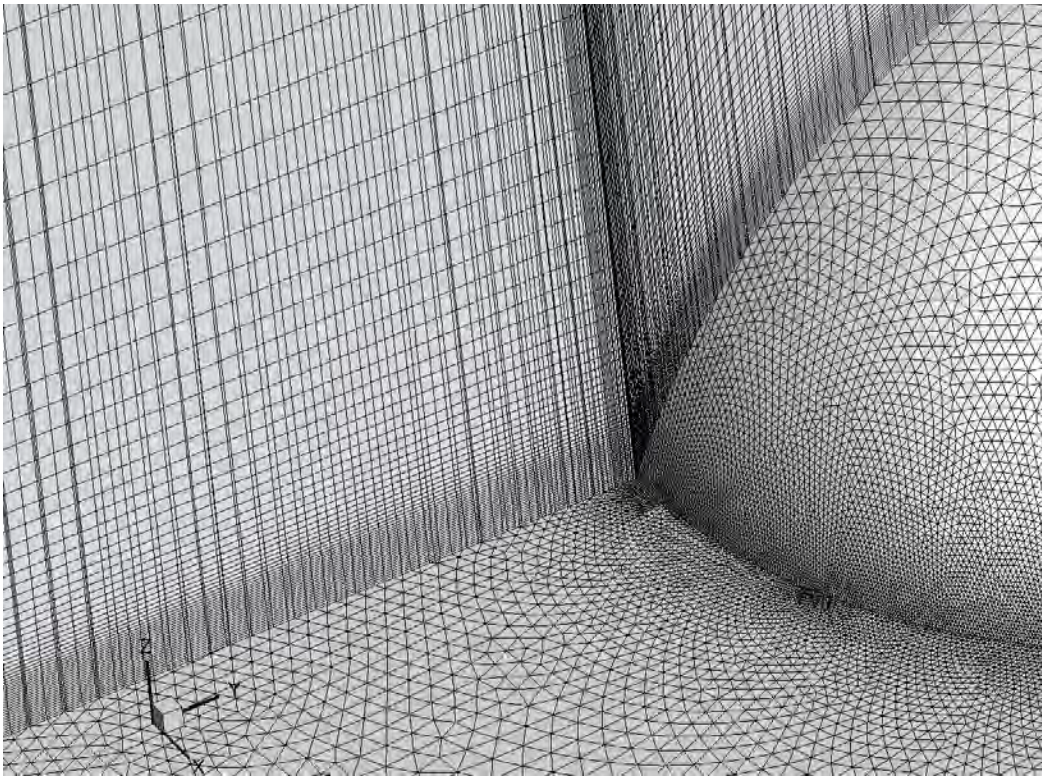

Figure 4.7. The triangulated surface grid of the ground and the hemisphere is shown for the geometry of the 6x7 array with a hemisphere. The vertical slice through the 'cooper mesh' shows the vertical spacing of the prisms as well as the skewness of the cells along the transition line from the ground mesh to the hemisphere mesh.

A quick grid study was done and it was found that a tetrahedral based mesh had better convergence qualities for the curved hemisphere topography; however, this technique has inadequate resolution for the boundary layer upwind of the container array. Therefore, a combination of these methods was produced.

This combination technique was only used for the hemisphere geometry since it was much more time consuming to build than the previous techniques. This mesh is called a hybrid unstructured mesh. This technique started similar to the cooper mesh from the ground up to 10 meters. The cooper mesh had a constant 0.25 meter resolution up to 5 meters and then had 11 cells stretching from an initial spacing of 0.25 from 5 meters up to 10 meters. The space around the hemisphere and above the cooper mesh was meshed with an unstructured tetrahedral mesh. Figure 4.8 shows a slice through the volume mesh of this hybrid unstructured mesh. The highly skewed and high aspect cells along the edge of the hemisphere are replaced by higher quality tetrahedrals.

A boundary layer on the hemisphere itself is not needed for this simulation, since this paper is primarily concerned with the bulk flow affecting the container array. A "transition" mesh from the cooper mesh to the tetrahedral mesh above it is needed to produce a high quality mesh. The large, high aspect ratio cells in the cooper mesh in the farbox have a height of 0.73 meters and a width of 30 meters. A tetrahedral cell created on top of this would create a cell that matches the 30 meter width, but would have a height of 30 meters as well. This would introduce a very large cell volume difference in a very short distance which can lead to numerical instabilities and poor convergence. A mesh that GAMBIT refers to as a "boundary layer" mesh was used as the "transition" mesh. This mesh is also comprised of triangular prisms like the cooper mesh, however it has more flexibility in it's growth. The "boundary layer" mesh in GAMBIT allows the mesh to grow from a given initial height to a final height that is based off of each cells aspect ratio in N number of cells. The height of this transition layer near the refined cells remains small while it retains good cell aspect ratios. It also allows the larger cells to grow at a faster rate to a more appropriate height based on each

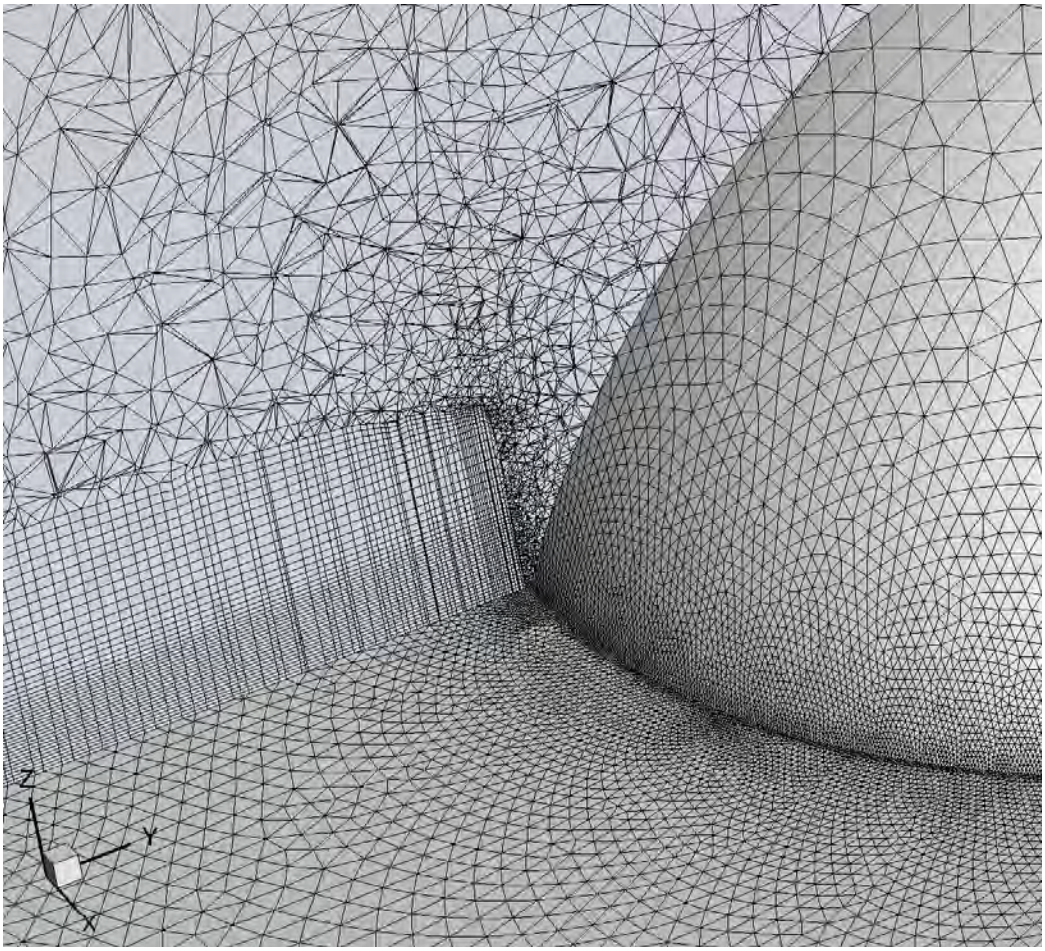

Figure 4.8. The triangulated surface grid of the ground and the hemisphere is shown for the geometry of the 6x7 array with a hemisphere. The vertical slice through the hybrid unstructured volume mesh shows the vertical spacing of the prisms in the 'cooper mesh' as well as the transition from the prisms to the tetrahedral cells.

cells final aspect ratio. A "transition" mesh was defined on the upper surface of the cooper mesh with a first row height of 0.73 meters to match the last layer of the cooper mesh, used a cell aspect ratio of 40% for the goal of the last layer of cells, and used 10 layers of cells as the transition mesh. The behavior of this "transition" mesh can be seen in Figure 4.9. The height of this mesh upwind of the array is larger than the height of the mesh around the container array. This

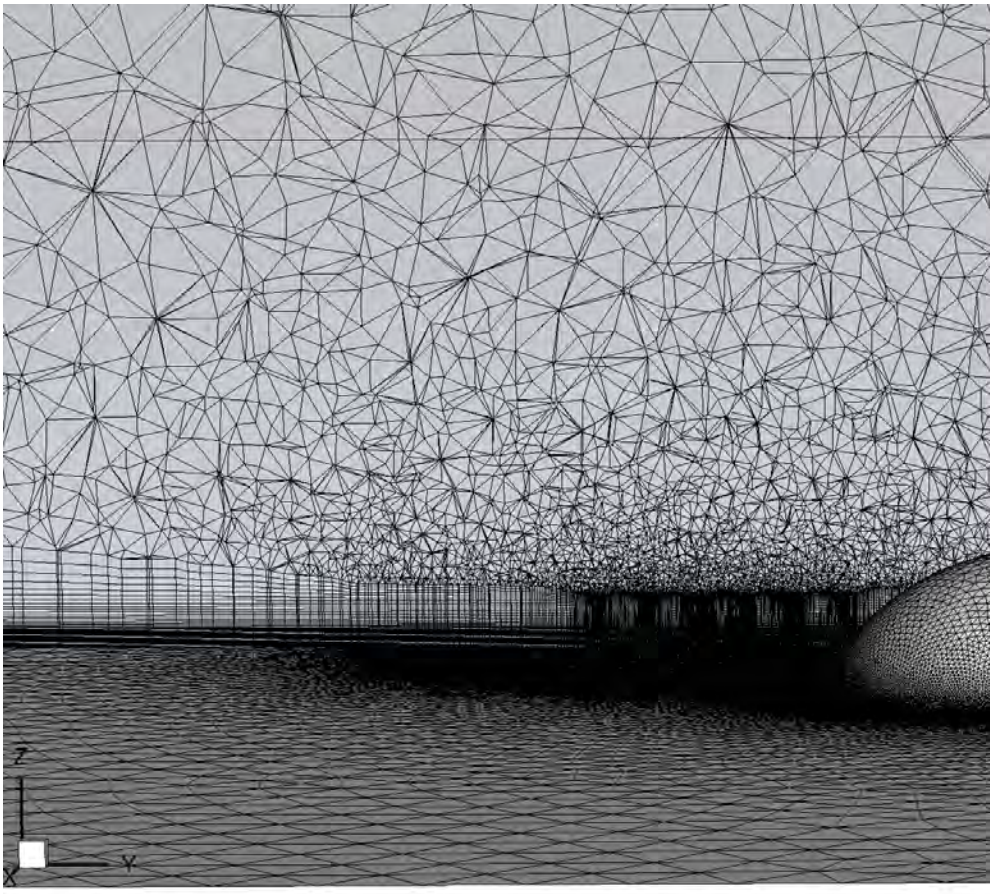

Figure 4.9. A wider view of the vertical slice through the hybrid unstructured volume mesh shows the growth in the height of 'transition' prisms cells as a function of the size of the triangulated surface grid on the ground. The triangulated surface on the ground grows as a function of increasing horizontal distance away from the hemisphere and 6x7 array similar to that shown in Figure 4.6.

transition layer creates a significantly better quality mesh than the previous one.

After this "transition" mesh was created the rest of the domain was filled in with unstructured tetrahedrals. This convergence for this mesh showed much improvement over the previous pure cooper mesh for the hemisphere geometry. To further improve the convergence, the top boundary was extended up to a height of 210 meters and the top of the domain had a symmetry boundary condition imposed to prevent backflow from coming back into the previous pressure-outlet boundary condition. This large spherical geometry is the reason that there is flow

leaving the top of the domain and also coming back in. The symmetry boundary basically acts as an inviscid wall and is believed to have a relatively negligible affect on the solution since it is over 100 'container heights' away. Quantifying this effect is a topic for further research. Once again this paper is primarily concerned with the mean wind field around the container array. This hybrid unstructured grid contained approximately 13.3 million cells.

It should be noted here that the process involved to create this cooper grid around this simple uniform array of containers took approximately 5 man hours to complete. This grid was built by a user with approximately three year of GAMBIT experience and approximately 7.5 years of experience with CFD in general. These grids were fairly simple and the geometry took relatively little time to model in GAMBIT, however the time to model a city quickly increases as the complexity increases. Previous experience with a more complicated city with 21 unique buildings took the same user more than 5 days (40 man hours) to grid and had roughly 70 million cells. Much of this time was spent in geometry preparation. Even this geometry would be considered relatively simple when compared to many of the urban areas throughout the world. The time to build the grid could also vary depending on the experience level of the user and the quality of the city geometry used. In an emergency, even 15 hours to build a simple grid might not be fast enough. The refinement level of the grid in CFD models also affects the result, so a poorly built grid could adversely affect the solution. QUIC-URB's empirically derived algorithms theoretically reduce the sensitivity of the grid refinement, however a study of the application of this theory should be done in future work. The grid used in QUIC is a simple cartesian grid that is "blanked" in the areas of the buildings. Once the urban city geometry is entered into QUIC and the x, y, and z spacing specified, the grid is done. By contrast, the 6x7 container array grid built with QUIC-GUI only took less than 20 minutes by a user with roughly 2 years of experience with QUIC-GUI.

4.3.2 CFD validation

The FLUENT mesh and solution methodology used in this paper should be compared to test data since the results from these computations will be used to validate QUIC-URB. This will loosely quantify the error involved by using a CFD solution as the input instead of test data. Future work should focus on gathering test data and using that as a direct comparison to the QUIC-URB data assimilation technique. This would bypass any error involved with using CFD, but would result in a smaller comparison to test data as mentioned in an earlier chapter.

The MUST experimental data is somewhat difficult to use as a comparison since it has temporally and spatially varying inflow conditions that are not simple to model. This is not to say that it is too difficult to model, but that it is out of the scope of this research and should be the focus of future work. A wind tunnel test of the MUST array [1] presents a great opportunity to compare the FLUENT SST-SAS model against an experiment with fewer unknowns and variables than the full scale MUST experiment.

Figure 4.10 shows the MUST wind tunnel test setup performed at the University of Hamburg in Hamburg, Germany [1]. The figure shows the MUST array at a 45° angle to the inflow with a uniform array of roughness elements upwind of the array.

A previous study [2] compared several MISKAM CFD solutions with this wind tunnel test data and will be used as a baseline when comparing the FLUENT SST-SAS solutions to the test data. The MISKAM solver is a microscale flow model that uses the RANS equations while adopting the Boussineq approximations to eliminate sound waves and uses a modified k-ϵ turbulence closure model. The MISKAM numerical flow model has gained a high level of acceptance by environmental agencies, consulting engineers and meteorologists as well as research institutes [15]. The MISKAM comparison [2] to the MUST wind tunnel test [1] concluded that the main flow features were resolved well; however, smaller structures in the vicinity of the containers were not correctly modeled, which is

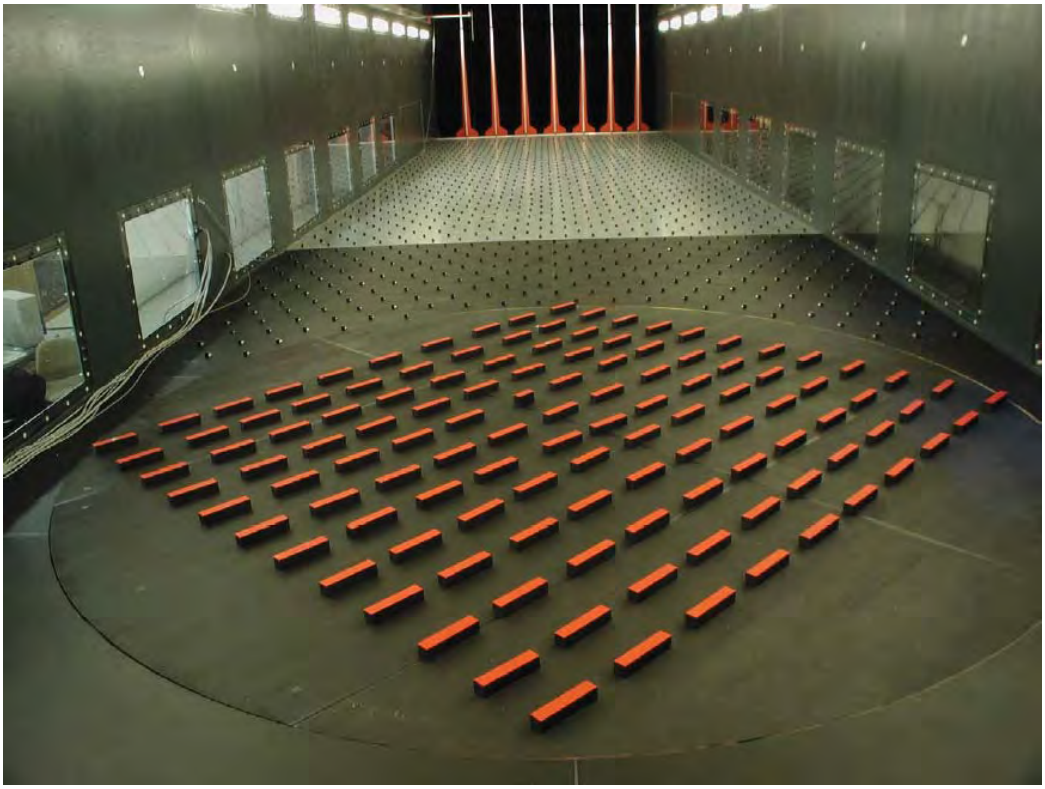

Figure 4.10. University of Hamburg wind tunnel test [1] of the MUST configuration. The direction of flow is coming from the guide veins (shown in the top center of the picture) over the small roughness elements and then through the scaled 10x12 MUST array.

in agreement with the previous observation [42] noted as one of the drawbacks of RANS models for meteorological flow. The main focus of this paper is to validate the data assimilation technique and how it captures the main flow features of spatially inhomogeneous flows this limitation of the RANS model will be acceptable. Future research should compare this wind tunnel data to an LES model.

The CFD analysis in the current paper closely follows inputs and suggestions made in the analysis [2] of the MUST wind tunnel test with a few exceptions. The slightly misaligned shipping containers from the fullscale MUST experiment was reproduced in the wind tunnel test, as shown in Figure 4.11, but was not modeled for this paper. The wall effects from the wind tunnel walls are assumed to be

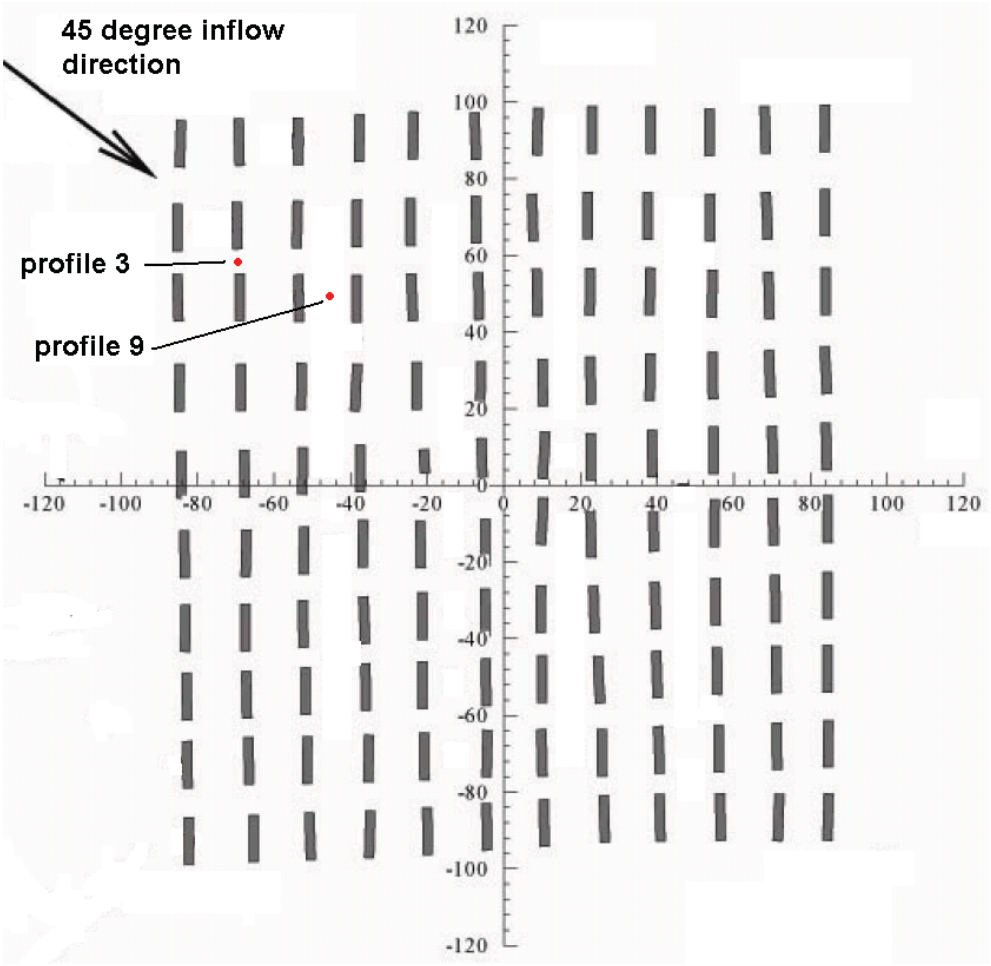

Figure 4.11. The plan view of the MUST wind tunnel test model [1] is shown with two of the profile measurement locations used for this validation. The arrow shows the direction of the incoming flow.

negligible near the centerline of the tunnel where the results will be compared so they are not modeled for this comparison. Future research should verify that these effects are negligible.

This FLUENT comparison used the 45° relative wind angle case as shown in Figures 4.10 and 4.11. Figure 4.11 illustrates the locations of the two vertical velocity measurements in the wind tunnel test. These measurements will be compared to the FLUENT and from the MISKAM [15, 2] simulation results.

The upstream inflow data gathered during the test suggests that a logarithmic

velocity profile given in equation 4.3 should be used.

$$u(z) = u_{ref}\left(\frac{ln\left(\frac{z}{z_o}\right)}{ln\left(\frac{z_{ref}}{z_o}\right)}\right) \tag{4.3}$$

where the reference velocity (u_{ref}) is 1 m/s, the reference height (z_{ref}) is 7.29 meters and the roughness height (z_o) is 0.01 meters.

The Balczó paper [2] indicates that the MISKAM simulations showed a sensitivity to the Turbulent Kinetic Energy (TKE) profile implemented at the inflow. The TKE inlet profile used is also based from the wind tunnel measurements [1].

Figure 4.12 shows a comparison of the measured velocity and TKE inlet profiles to the inputs used for the MISKAM and FLUENT simulations. The FLUENT TKE inlet profile was approximated using a sectional polynomial fit. Fortunately, the sharp kink in the profile is numerically smoothed out by the TKE dissipation used by the solver before the flow reaches the container array. A MUST container was added to the plots for reference.

The FLUENT simulation ran on three grids with different levels of refinement and two different grid methodologies. A rigorous grid sensitivity study was not performed since the previous MISKAM study had provided an adequate starting point. However, further research should investigate if the FLUENT SST-SAS model was properly grid resolved by starting with a grid resolution that was verified by a RANS model.

A hybrid unstructured mesh and a cooper mesh were the grid methodologies used. A horizontal mesh resolution of 0.5 and 0.25 meters within the container array were the two levels of refinement used for the cooper mesh. Both meshes had a 0.25 meter vertical spacing up to 5 meters with 0.5 and 0.25 meter resolution meshes containing approximately 13.2 and 29.7 million cells, respectively. The hybrid unstructured mesh had horizontal resolution of 0.25 meters within the array and 0.25 meter resolution in the vertical direction up to a height of 5 meters as well. Inside the wakebox and within the container array the mesh consisted of tetrahedral. Outside the wakebox was a cooper mesh to properly capture the boundary layer on the ground upwind of the container array. This mesh had

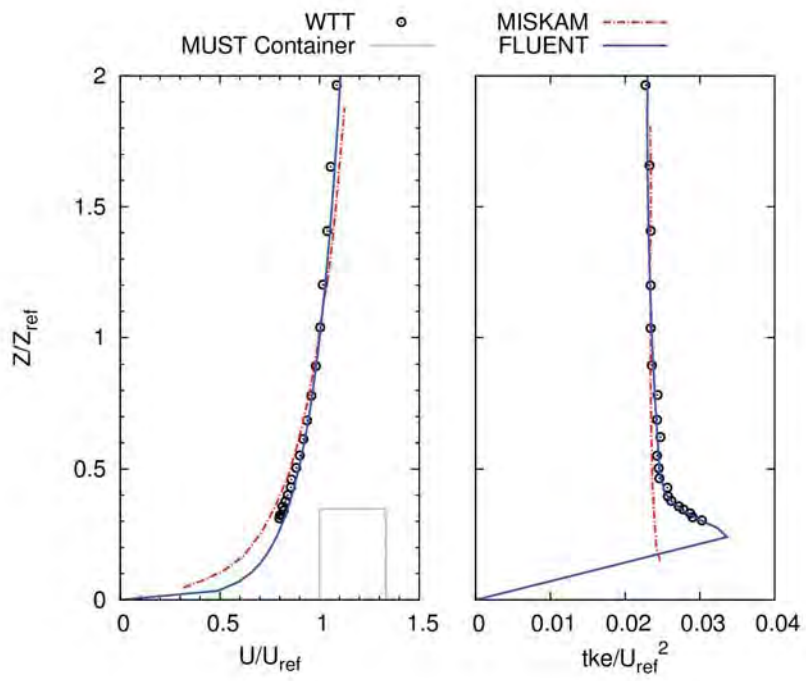

Figure 4.12. Normalized vertical velocity and turbulent kinetic energy profile comparisons at the inlet for the wind tunnel test [1], MISKAM simulation [2], and the FLUENT simulation are compared. A container was also plotted to give a reference scale. $u_{ref} = 1$ m/s, $z_{ref} = 7.29$ meters

a total of 31.7 million cells. Each mesh used a similar set of size functions as mentioned earlier in Section 4.3.1.

Figures 4.13 and 4.14 show the comparison of these three grids at the two measurement profile locations illustrated in the lower left caption of each of these figures and also illustrated earlier in Figure 4.11.

These figures also include the results from the MISKAM simulation [15, 2] for reference. A container was also plotted to show the scale of the solutions. The velocity plots were normalized by a z_{ref} of 7.29 meters and a u_{ref} of 1 meter/sec since that is what was used in the test. It is difficult to choose which grid to use from looking at these two figures since there is not one that clearly outperforms

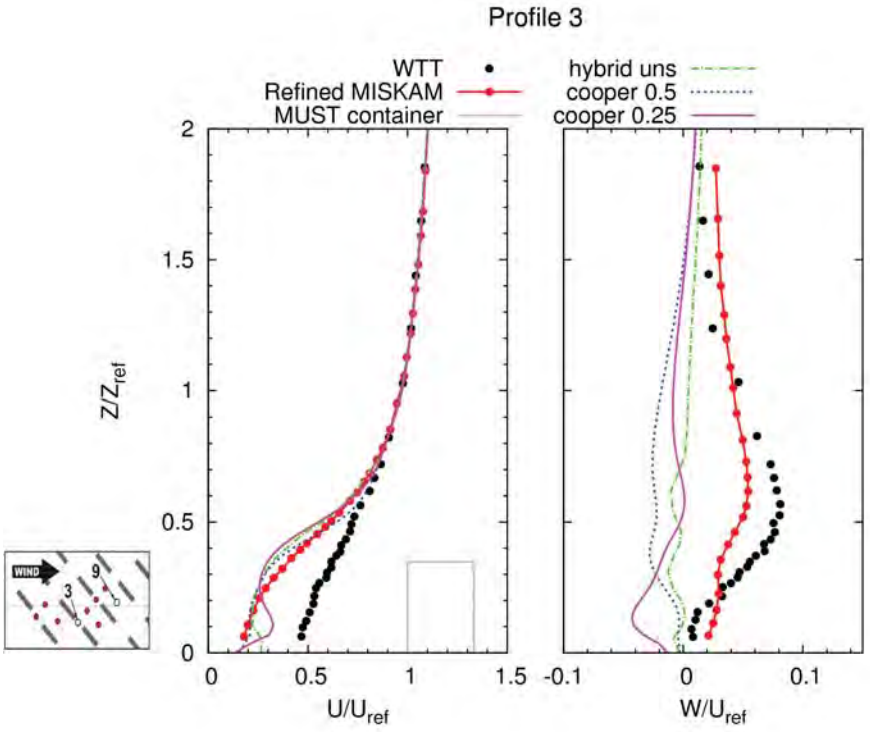

Figure 4.13. Normalized vertical U and W velocity profile comparisons of the wind tunnel test [1], the refined MISKAM simulation [2], and the three following types of grids used at location 3 of the wind tunnel test shown in the inset: a hybrid unstructured grid, a 'cooper grid' constructed with triangulated prisms with a 0.5 meter horizontal refinement and another 'cooper grid' with a 0.25 meter horizontal refinement. $u_{ref} = 1$ m/s, $z_{ref} = 7.29$ meters

the others everywhere. The refined cooper mesh 'cooper 0.25' matches the trend of the vertical velocity (W) profile of the test data slightly better than the other two grids at profile 3, however it has a strange horizontal velocity (U) fluctuation near the ground. This refined cooper mesh also showed a larger negative magnitude of vertical wind speed near the top of the building at profile location 9 which resembled the test dat even though the trend was shifted up from the tunnel data. From this small amount of data the refined cooper mesh 'cooper 0.25' was chosen as the baseline mesh for the QUIC-URB validation. Further research should compare data from additional measurement locations as well as data from

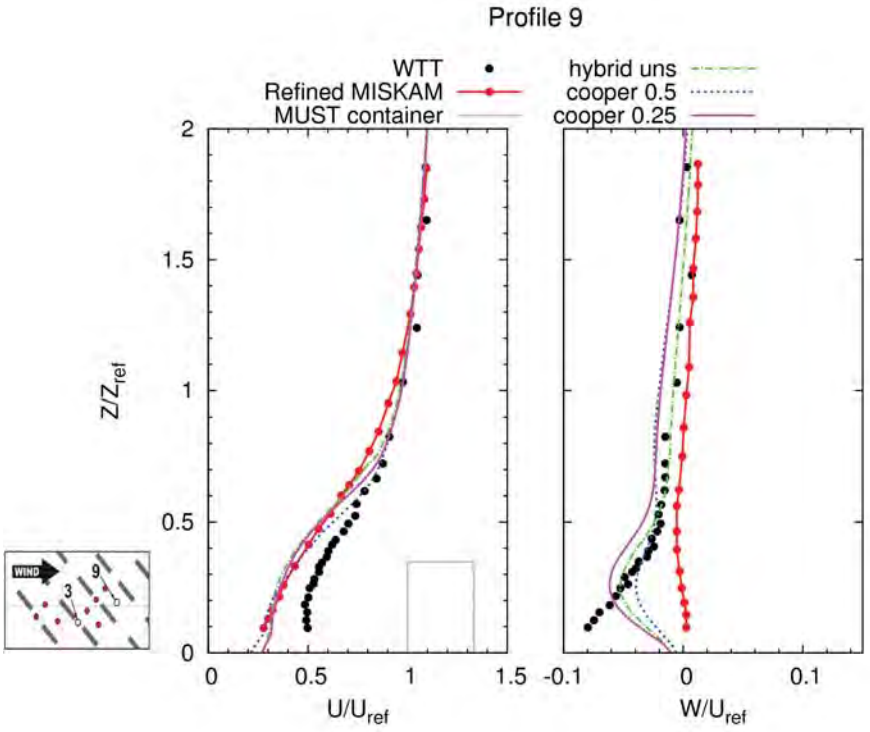

Figure 4.14. Normalized vertical U and W velocity profile comparisons of the wind tunnel test [1], the refined MISKAM simulation [2], and the three following types of grids used at location nine of the wind tunnel test shown in the inset: a hybrid unstructured grid, a 'cooper grid' constructed with triangulated prisms with a 0.5 meter horizontal refinement and another 'cooper grid' with a 0.25 meter horizontal refinement. $u_{ref} = 1$ m/s, $z_{ref} = 7.29$ meters

other angles of inflow taken during this test to additional FLUENT SST-SAS simulations to further validate the proper mesh resolution and input conditions.

4.3.3 FLUENT inputs

The inputs and grids described in the previous sections are used to compute the FLUENT solutions that will be used as inputs to QUIC-URB then compared to the QUIC-URB solutions. These inputs were compiled into an input file called a 'journal' file. A FLUENT journal file has all the information needed to run a full FLUENT simulation. A FLUENT journal file was used to set the boundary

conditions, the physics models, the numerical methods, the initial conditions and it also exported the convergence data for each case. The journal file for the baseline 6x7 array is in Appendix B. Any variables not specified in this journal file are set to FLUENT's default value. The other two geometries used a very similar journal file modified slightly to account for slightly different boundary conditions arising from the different geometry used.

A high performance cluster (L1.jsc.nasa.gov) at NASA Johnson Space Center was used for all of the FLUENT analyses. L1 is an SGI Altix ICE 8200LX computing cluster consisting of 1344 2.67Ghz Westmere (X5650) processors. These resources are available to batch jobs using the PBS batch system. The PBS script used to submit the FLUENT batch jobs is located in Appendix B.

The meteorological inputs of the three geometries analyzed in this paper (6x7 array baseline, 6x7 array with a hemisphere, and 6x7 array with a wall) are identical to the inputs used to validate the CFD solution in section 4.3.2. These inputs were used since they were representative of the fullscale MUST experiment [8].

The boundary condition used for the top, sides and downwind faces of the domain is the pressure-outlet boundary condition, while the upwind face boundary condition is set to a velocity inlet condition. This velocity inlet condition is defined by an User Defined Function (UDF) that is compiled beforehand. The UDF specified the vertical velocity profile from equation 4.3 and the TKE inlet profile shown in Figure 4.12 via a sectional polynomial fit.

Each case is initialized with a constant 1 m/s velocity in the streamwise (Y) direction, then three levels of Full Multi-Grid (FMG) initialization is performed to quickly get a rough approximation of the answer on a succession of grids with much fewer cells than the full mesh. After FMG initialization a steady-state solution with the $k-\omega$ SST turbulence model is run for 3,000 iterations or until converged. From this steady state convergence the solutions are checked for any temporal sensitivities by running a dual-time accurate simulation for at least 1,000 iteration to see if the solution changes. In the case of the hemisphere with the hybrid mesh

it was run for a total of 11,000 iterations to check for adequate convergence. There are eight subiterations per time-step and the Scale Adaptive Simulation (SAS)-SST turbulence model is used during this time accurate simulation. The PBS script that resubmitted the FLUENT batch job and started the time accurate simulation is shown in Appendix B.

4.3.4 FLUENT convergence

It is necessary to know the proper time to stop the iterative process of solving the governing equations when using CFD to solve a problem. The solution will reach a point when it is sufficiently converged and therefore the quantities of interest in the given solution will no longer change significantly with additional iterations.

One indication that the solution has converged is the residual history. The residual is the imbalance in the equations being solved. For example, after discretization, the conservation equation for a general variable ϕ at a cell P can be written as

$$a_P \phi_P = \sum_{nb} a_{nb} \phi_{nb} + b, \tag{4.4}$$

where a_P is the center coefficient, a_{nb} is the influence coefficients for the neighboring cells, and b is the contribution of the constant part of the source term S_c in $S = S_c + S_P \phi$ and of the boundary conditions. The center coefficient a_P is defined as

$$a_P = \sum_{nb} a_{nb} - S_P. \tag{4.5}$$

The residual R^ϕ computed by FLUENT's pressure-based solver is the imbalance in equation 4.4 summed over all the computational cells P [44]. This is referred to as the "unscaled" residual and is written as

$$R^\phi = \sum_{cells P} \left(\sum_{nb} a_{nb} \phi_{nb} + b - a_P \phi_P \right). \tag{4.6}$$

In general, it is difficult to judge convergence by examining the residuals defined by equation 4.6 since no scaling is employed. Therefore, the residuals plotted in this report were "locally scaled" residuals and defined as

$$R_\phi = \frac{\sqrt{\sum_{cells}^{n} \left(\frac{1}{n}\right) \left(\frac{\sum_{nb} a_{nb}\phi_{nb} + b - a_P\phi_P}{a_P}\right)^2}}{(\phi_{max} - \phi_{min})_{domain}}. \tag{4.7}$$

For the momentum equations the denominator term $a_P\phi_P$ is replaced by $a_P v_P$, where v_P is the magnitude of the velocity at cell P. This scaled residual is a more appropriate indicator of convergence for most problems and therefore was used in this study.

Three Orders of Magnitude (OOM) drop of the residuals of each of the equations being solved is considered good practice. The residuals should also be steadily decreasing or, at a minimum, leveled off (i.e., they should not be increasing). The forces of the ground, buildings and other objects should also be monitored as another indication of solution convergence. Once the integrated pressure on these surfaces is no longer changing in time the force history plot will level off. However, an oscillatory or chaotic pattern of this force history plot is an indication that the solution is unsteady. If the force history oscillates around a steady mean value then this will be as far as a steady state solution can converge the problem. At this state it is possible to use an unsteady solver without changing the solution by any significant amount. However, it is also possible that this unsteadiness is numeric, so the solution may change with a time accurate numerical scheme. Therefore, for this paper this temporal sensitivity was checked for each of the three cases to make sure the solutions were properly converged. This time accurate solution is also the point that the SST-SAS turbulence model was employed.

Figures 4.15 through 4.20 show the force and residual convergence of each of the three different geometries. Figure 4.15 shows the unsteadiness and chaotic nature of the total force acting on the ground during the steady-state portion of the baseline geometry solution. At 3,000 iterations the time accurate simulation

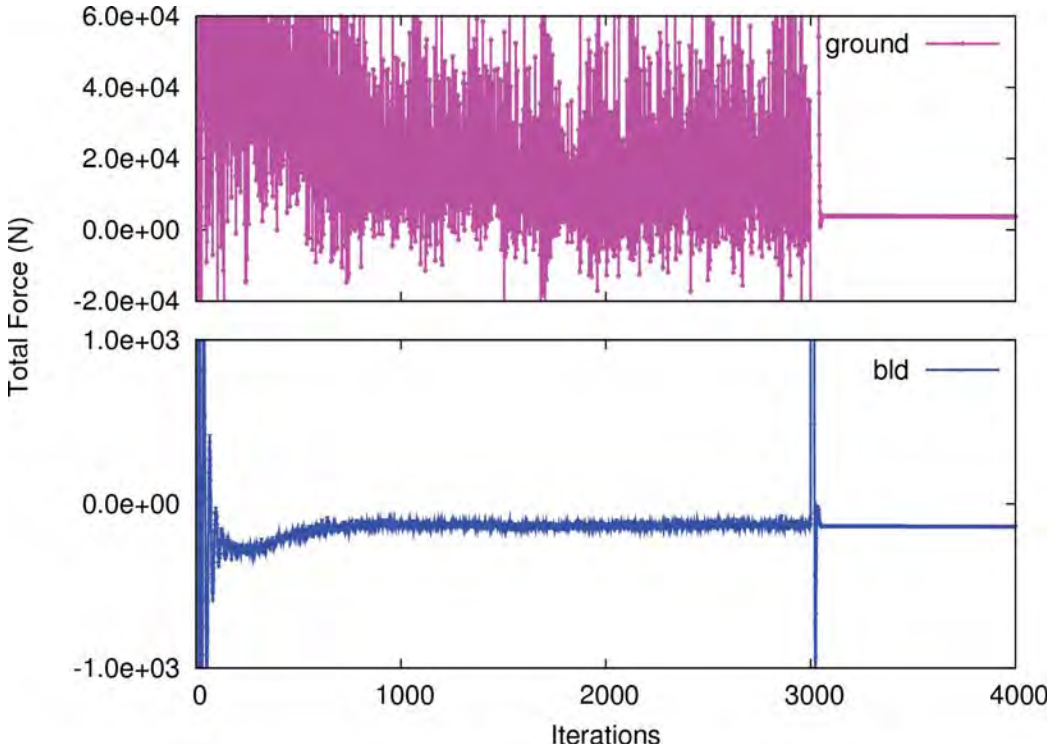

Figure 4.15. The total force convergence history for ground (upper plot) and the containers (lower plot) of the baseline 6x7 array geometry starting with the steady state solver and then switching over to a time-accurate scheme at 3,000 iterations.

with the SAS turbulence model was started. This dual-time accurate scheme and advanced turbulence model steadied the force out to a smooth level line. The unsteadiness in the force history was most likely due to numerical unsteadiness.

Figure 4.16 shows roughly a 3 OOM drop of the scaled residuals for each of the model equations during the steady state solution. During the time accurate solution the smallest residual drop is an additional 2 OOM. Each equations horizontal line in the time-accurate residual history represents each of the eight subiterations. The total residual drop for each iteration is measured by the lowest subiteration point. For example, the ω equation in black drops down to roughly 1.0e-09 at 3,700 iterations, so this is a 9 OOM drop of the scaled residual at this iteration for the ω equation.

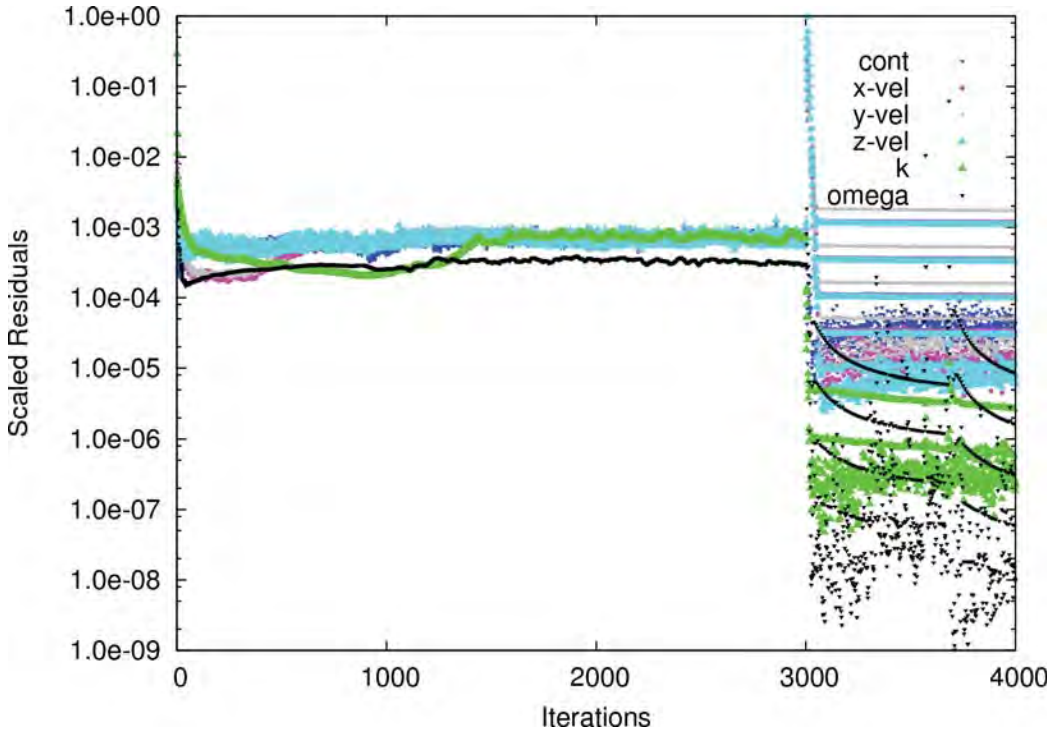

Figure 4.16. The convergence history of the scaled residual as given by equation 4.7 for the baseline geometry and showing the steady state solution being switched over to the time-accurate scheme at 3,000 iterations.

Figures 4.17 and 4.18 show the convergence histories of the forces and scaled residuals of the wall geometry which displays a similar convergence trend to the baseline case. This solution does not have as much numerical unsteadiness as the baseline solution.

This case was not switched to the time accurate scheme until 4,000 iterations since the slope on the ground force history at 3,000 iterations was still trending down. The scale residual convergence for this case is also much more smooth and dropped down about 5 OOM for the z-velocity equations.

Figures 4.19 and 4.20 show the convergence histories of the forces and scaled residuals of the hemisphere geometry which displays a very similar convergence trend to that of the wall case. The time-accurate portion of the simulation was also started at 4,000 iterations and was carried out until 11,000 iterations to make

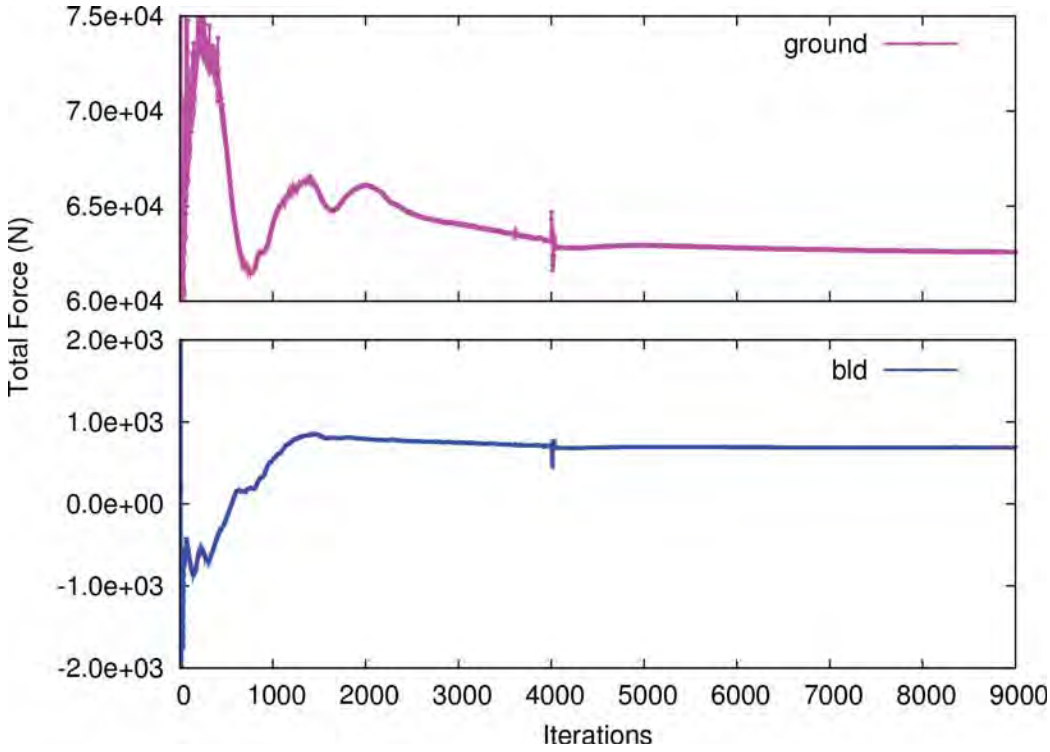

Figure 4.17. The total force convergence history for ground (upper plot) and the containers (lower plot) of the 6x7 array with a wall starting with the steady state solver and then switching over to a time-accurate scheme at 4,000 iterations.

sure it was completely converged since previous attempts had shown unsteadiness and diverged after many iterations. All of these cases reached an acceptable level of convergence.

4.3.5 Postprocessing

The FLUENT results were postprocessed in order to easily compare the velocity vector field with the results from the QUIC-URB solutions. This process removed the FLUENT data beyond the range that QUIC-URB calculated. The FLUENT solutions where then interpolated onto the QUIC-URB grid in order to calculate the NRMSE and plot the difference contours in the velocity vector fields.

All of the postprocessing scripts are located in Appendix D. A summary of the steps involved in postprocessing the FLUENT solutions to extract inputs for

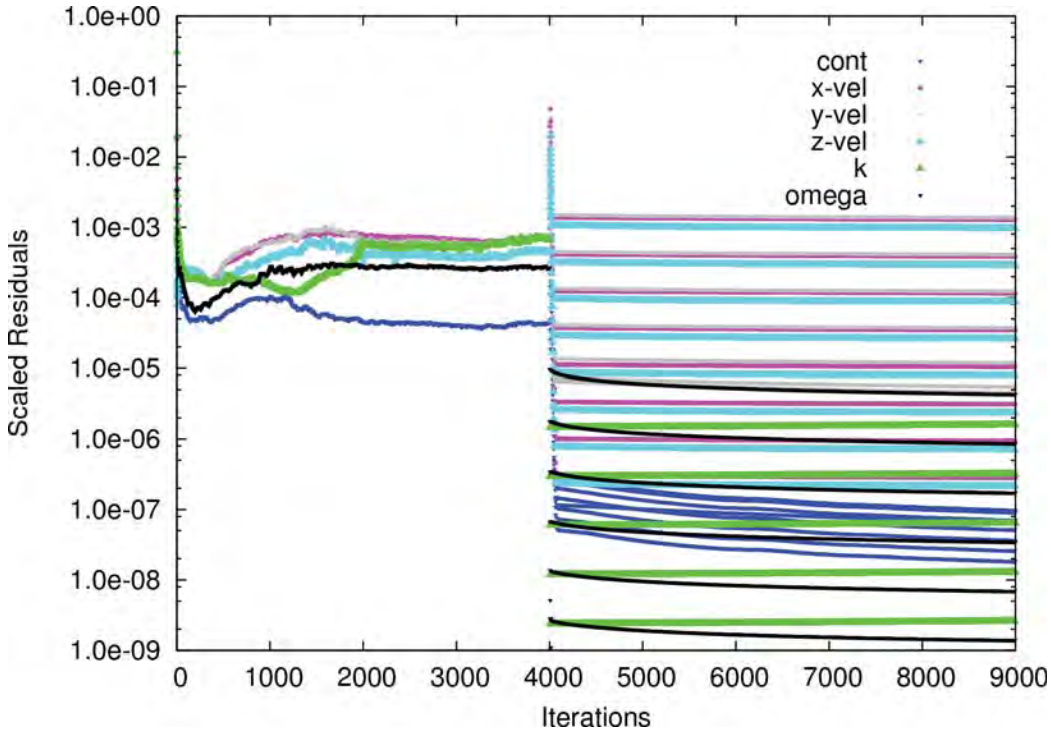

Figure 4.18. The convergence history of the scaled residual as given by equation 4.7 for the 6x7 array with a wall and showing the steady state solution being switched over to the time-accurate scheme at 4,000 iterations.

the QUIC-URB problems and compare the differences between the two methods follows:

1. The vertical velocity profiles to be imported into QUIC-URB where extracted using the Tecplot360 macro 'dump.mcr' in Appendix D.2.

2. The exported tecplot velocity profiles were converted into a plain txt file file using a perl script (tec2txt) in Appendix D.3.

3. These text files were converted into the quic input format to be read into QUIC-URB with the 'datastrip_quic' perl script in Appendix D.4 These files were then transferred to the QUIC-URB project directories.

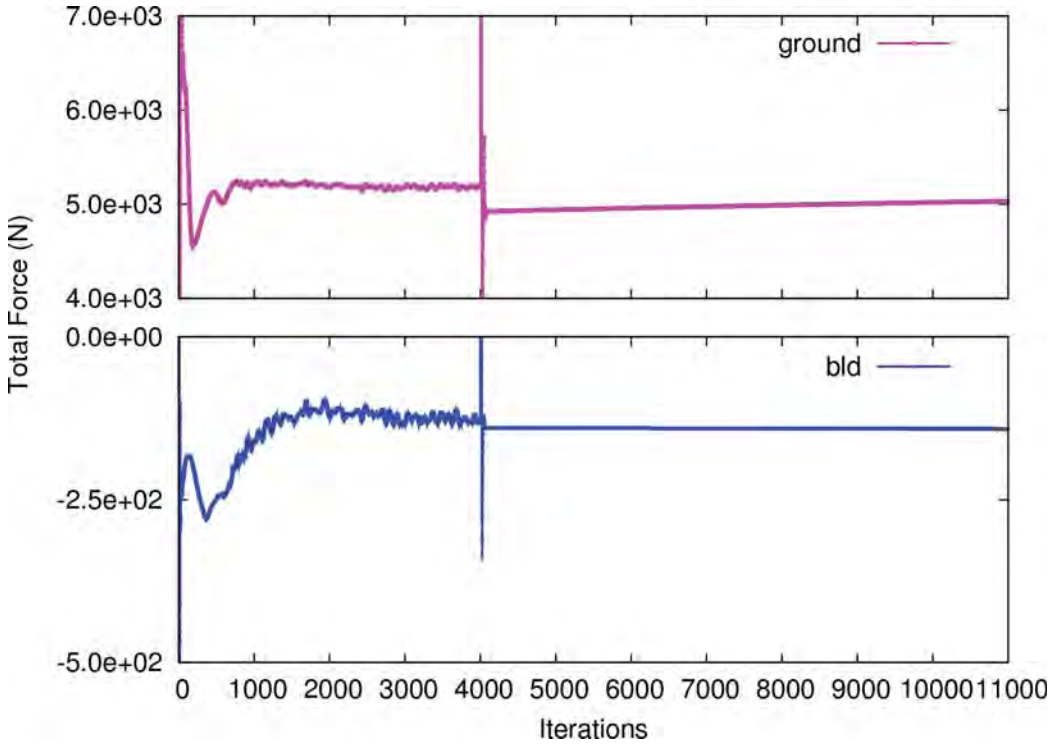

Figure 4.19. The total force convergence history for ground (upper plot) and the containers (lower plot) of the 6x7 array with a hemisphere starting with the steady state solver and then switching over to a time-accurate scheme at 4,000 iterations.

4. The FLUENT solution files were loaded into Tecplot360. The solutions were averaged to the nodes by Tecplot's arithmetic averaging technique. The X, Y, Z, U, V, W variables were then exported to a tecplot data file using the tecplot macro in Appendix D.1.

5. The tecplot files were converted to plain columnized text files with the 'nodecentertec2txt.csh' csh script listed in Appendix D.5.

6. The file size of the FLUENT vector velocity field was reduced by removing data beyond the outer bounds of the QUIC-URB domain using the Matlab 'FLUENT_file_reduction.m' function in Appendix D.6. This file also trans-

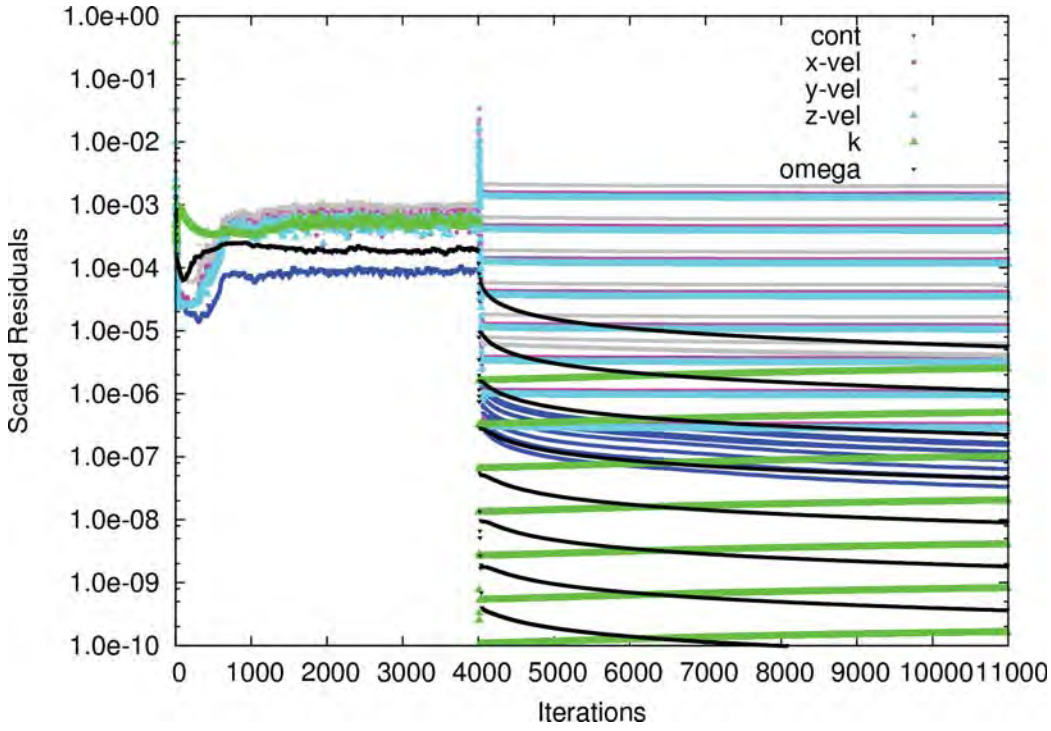

Figure 4.20. The convergence history of the scaled residual as given by equation 4.7 for the 6x7 array with a hemisphere and showing the steady state solution being switched over to the time-accurate scheme at 4,000 iterations.

lates the FLUENT solution to align its coordinate system to the QUIC-URB coordinate system. x=x+160, y=y-200;

7. This reduced FLUENT text file was sorted into the QUIC-URB ordered format using the 'fluent_file_sort.m' function in Appendix D.7.

8. All duplicate points from the ordered FLUENT file were stripped out using the Perl script 'unique_data.pl' given in Appendix D.8.

9. The size of some of the FLUENT files were too large for the Matlab interpolation function in the next step so all of the files were split up into 1.5 meter vertical sections using the 'fluent_file_reduction_zlimit.m' function in the Appendix D.10. This also enabled the analysis of the Normalized Root Mean Square Error (NRMSE) as a function of height.

10. The FLUENT data was interpolated to the structured QUIC-URB mesh domain using the 'fluent_interp.m' function in Appendix D.11. This file uses the 'read_windfield.m' function in Appendix D.12 to read the QUIC-URB grid. A batch script in Appendix D.13 was used to call the interpolation script for each of the FLUENT solution files.

11. The Matlab function 'compare_flow_fields.m' in Appendix D.14 calculates the difference between the QUIC-URB solution and the FLUENT solution. It also calculates the NRMSE for each vertical section and creates a series of error contour plots. A batch script 'batch_compare.m' in Appendix D.15 calls 'compare_flow_fields.m' iteratively for each set of QUIC-URB solution files.

Matlab version 7.10.0.499 (R2010a) was used to execute the Matlab scripts and Tecplot 360 2010 Release 1, Build 12.2.0.9077 for LINUX (64-bit) was used for the FLUENT data extraction.

4.4 QUIC-URB solution

This chapter describes the input files (Appendix C) used during the QUIC-URB data assimilation runs. The domain of the QUIC-URB simulations is 320x727x40 meters in the X, Y, and Z directions respectively. The resolution of the mesh is 1x1 meter in the horizontal directions and 0.5 meters in the vertical direction. The details of the simulation parameters used for each case is given in Appendix C.1, while the geometry for the 6x7 container array is given in Appendix C.3.

Nine profiles were extracted from the FLUENT solutions and are shown in the QUIC-URB coordinate system displayed in Figure 4.21. The distances on the top and sides of this figure are the x and y coordinates the data profiles extracted. This image is not to scale. The QUIC-URB coordinate system differs from the FLUENT coordinate system and should not be confused with each other. For reference, the FLUENT coordinate system was shown previously in Figures 4.3 and 4.4. The proper coordinate conversion was done when extracting the profiles from FLUENT and inputting them into QUIC-URB.

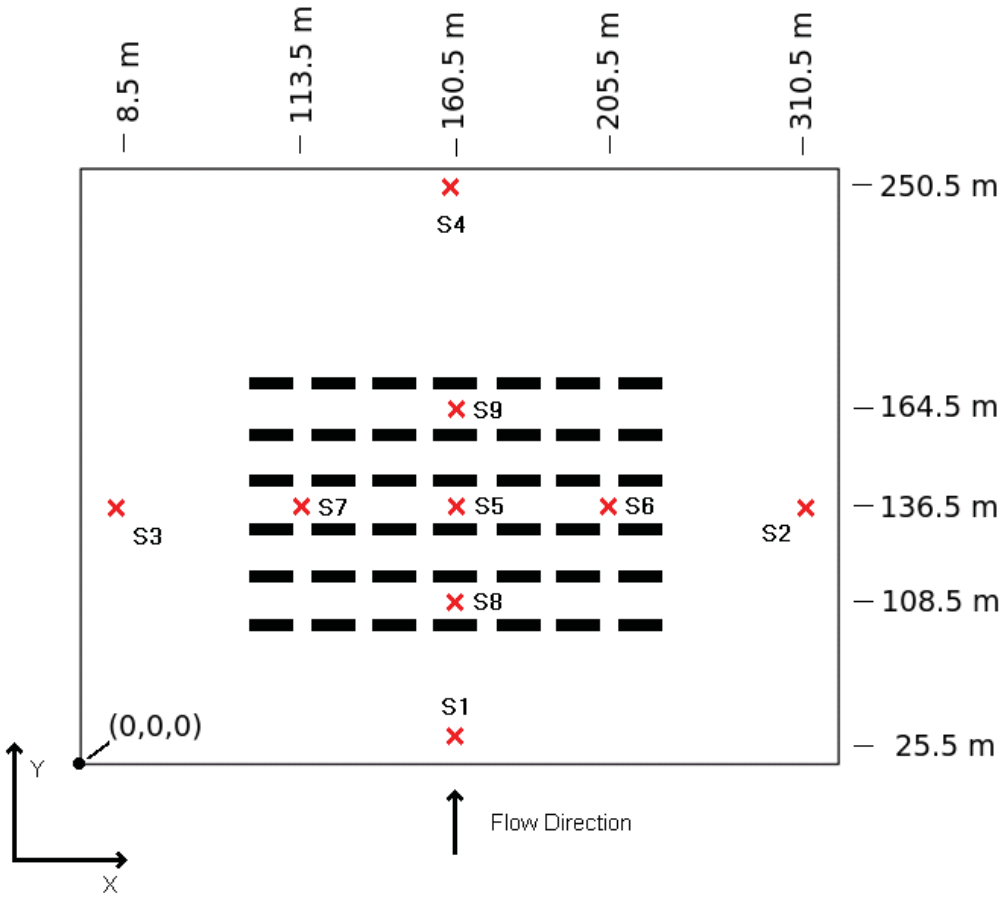

Figure 4.21. Illustration of the "sensors" in and around the container array extracted from the FLUENT solution and ingested into QUIC-URB. Each one is labeled and the measurements on the side of the axes show the coordinates of each row and column of the sensors.

The first four profiles were distributed evenly outside the container array. The first three should not see any effects from the baseline array at all. The 4^{th} profile is inside the wake of the container array. Profile 5 is in the middle of the array between buildings in the recirculation zone, while profiles 6 and 7 are located directly in the street canyon without any buildings directly upwind. Profiles 8 and 9 are located in the recirculation zones at the leading and trailing edge of the building array. The locations of these profiles were chosen to show the strengths and weaknesses of this new data assimilation technique.

There were nine QUIC-URB simulations performed for each of the three FLUENT geometries. The first QUIC-URB simulation for each of the geometries used only the profile from location 1. This simulation represents the QUIC-URB simulation result without using a data assimilation technique. The next simulation used profiles 1 and 2. The third simulation used profiles 1 through 3 and so on until the 9^{th} simulation which used all nine profiles.

The meteorological conditions are defined for each of the different cases using the file shown in Appendix C.2. This file is the QUIC-URB input file that imported all nine velocity profiles extracted from the FLUENT simulation. Each of these profiles can be seen in Figures 4.22 through 4.24, while an example of the input QUIC-URB format for each of the velocity profiles is shown in Appendix C.7.

This assimilation technique ignores the vertical velocity (W) which is a significant drawback to this method. For a reference of the magnitude of the vertical velocity data not being used the vertical velocity is also shown in the figures below. Ignoring this velocity component affects the vertical advection of the fluid and could affect the QUIC-URB solution. Future research should focus on adding the vertical velocity component to the data assimilation technique to quantify the size of this effect.

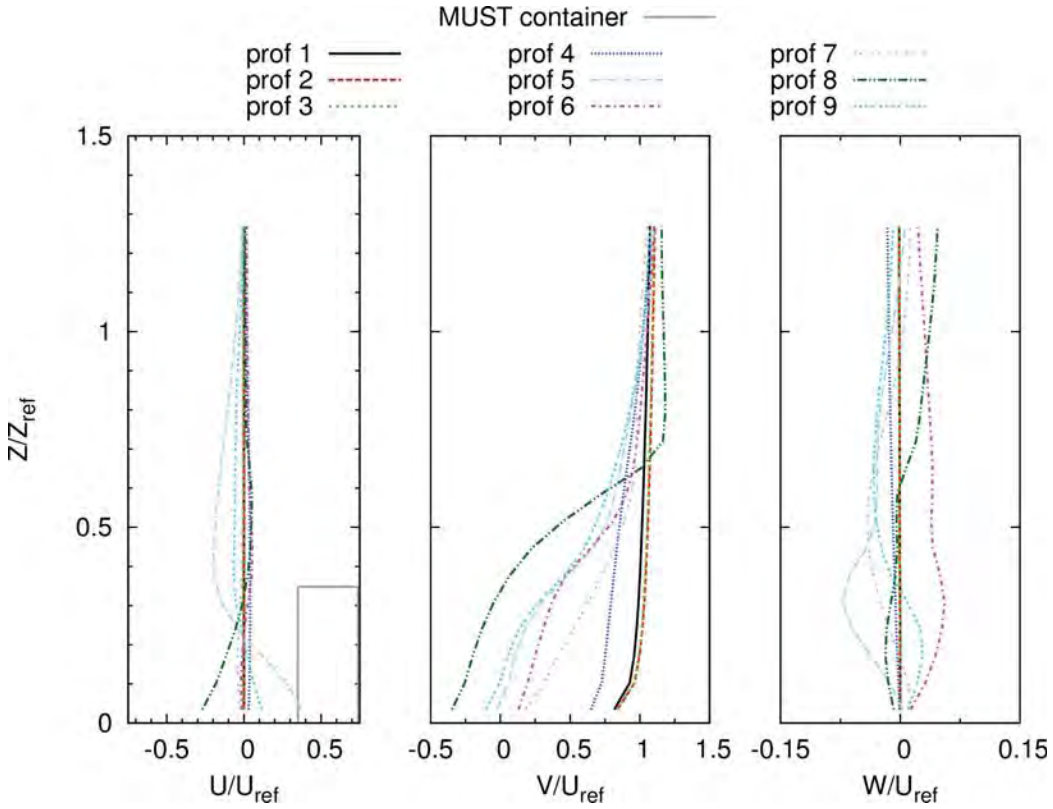

Figure 4.22. Velocity profiles extracted from the FLUENT solution of the baseline 6x7 array at each of the nine "sensor" locations, where $u_{ref} = 1$ m/s and $z_{ref} = 7.29$ meters.

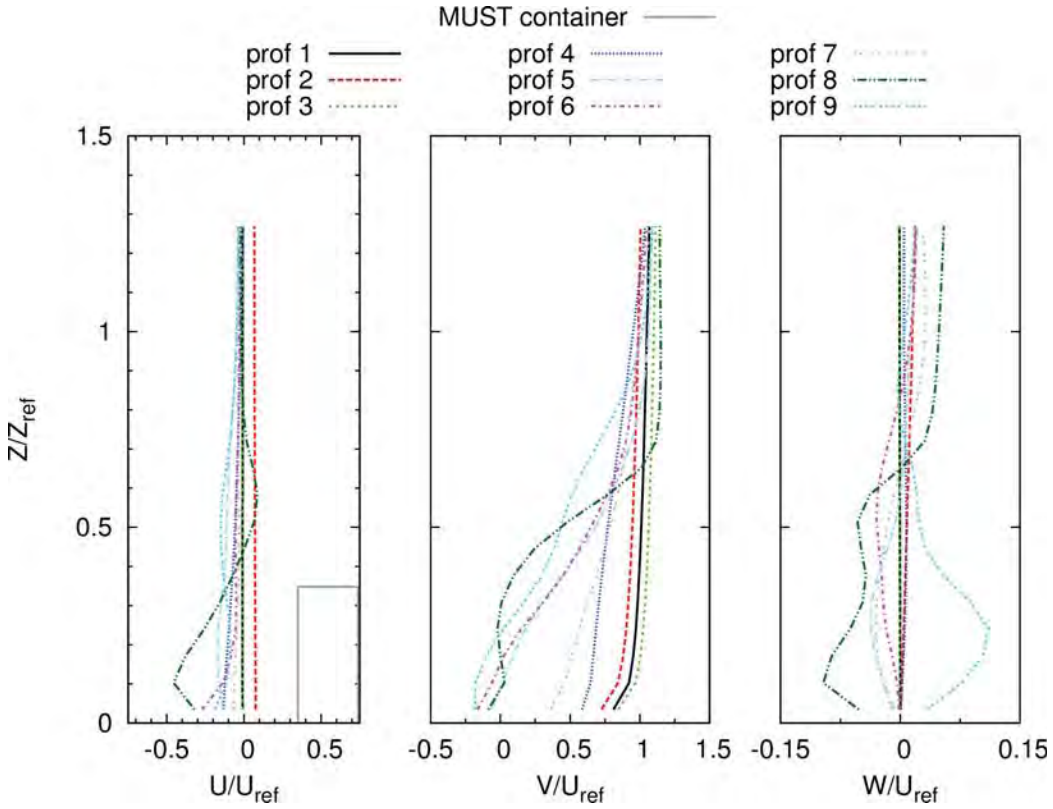

Figure 4.23. Velocity profiles extracted from the FLUENT solution of the 6x7 array with a hemisphere at each of the nine "sensor" locations, where $u_{ref} = 1$ m/s and $z_{ref} = 7.29$ meters.

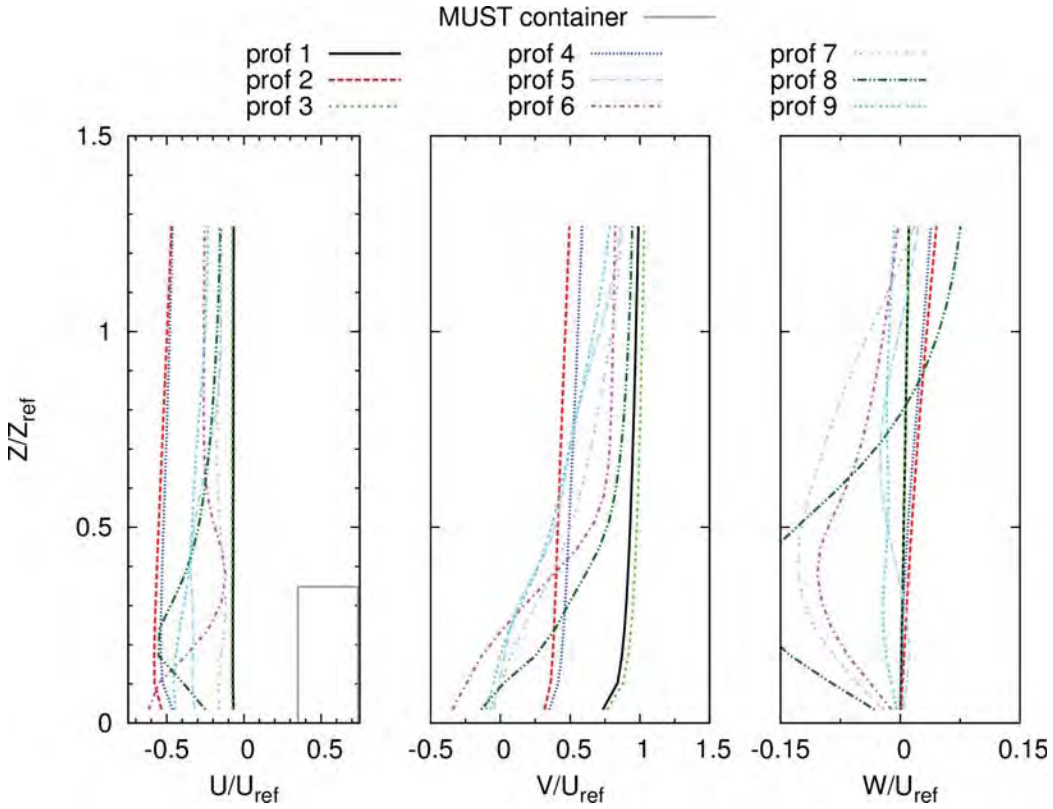

Figure 4.24. Velocity profiles extracted from the FLUENT solution of the 6x7 array with a wall at each of the nine "sensor" locations, where $u_{ref} = 1$ m/s and $z_{ref} = 7.29$ meters.

CHAPTER 5

RESULTS

This chapter contains the qualitative and quantitative results from the direct comparison of the FLUENT CFD solutions of the three topographies to the 18 QUIC-URB solutions utilizing the velocity profiles extracted from these FLUENT solutions.

The qualitative results show the velocity component differences. The U, V, and W QUIC-URB velocity components are subtracted from the respective FLUENT velocity components. Constant Z contour slices of these differences are visualized to highlight the largest differences. The contour slices provide a qualitative look at the strengths and weakness's of the data assimilatioin technique implemented in the latest QUIC-URB model.

The quantitative results for the nine QUIC-URB simulations corresponding to each of the three FLUENT topographies are calculated with the Normalized Root Mean Square Error (NRMSE) given earlier in equation 4.2. The domain was split into four 1.5 meter vertical sections. The first section started at the ground and extended up to 1.5 meters. The second section started at 1.5 meters and went to 3.0 meters, while the 3^{rd} and 4^{th} 1.5 meter sections ended at 4.5 and 6.0 meters respectively. The NRMSE of the U, V, and W velocity components and the velocity magnitude was calculated for each vertical section. Splitting the domain into these vertical sections provide some insight into the error as a function of height above the ground.

FLUENT solution visualizations of the flow through and around the urban canopy were produced to give the reader an insight into the flow patterns of the three topographies used in this analysis. Figures 5.1 through 5.3 show streamlines

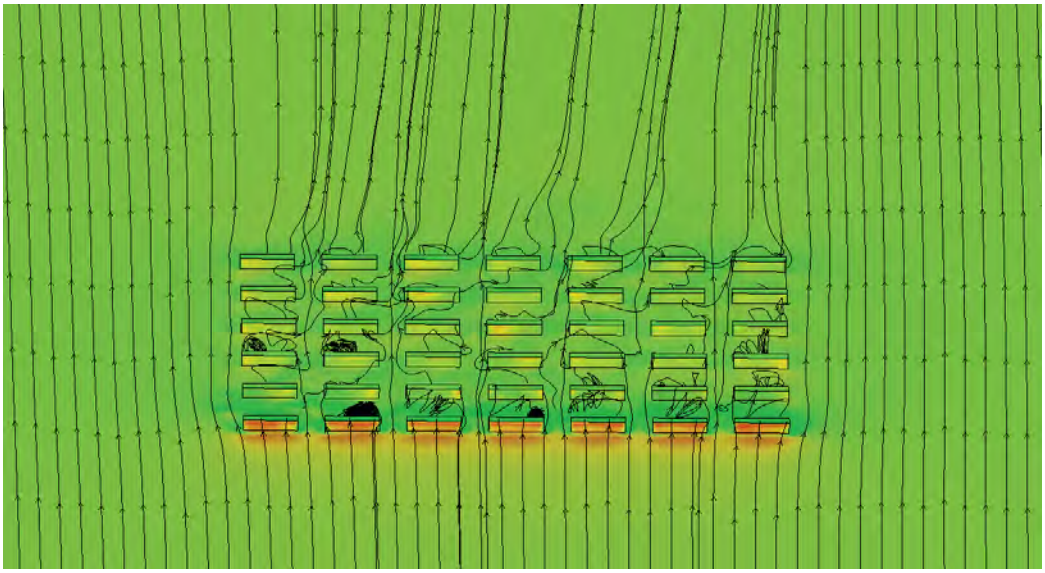

Figure 5.1. Streamlines and surface pressure contours for the baseline 6x7 array case from the FLUENT solution.

at a height of 1.25 meters (half the height of the containers). The ground, containers and walls have contours of pressure.

5.1 Comparison with a hybrid-RANS technique

The following subsections focus on the three different topographies analyzed. The first section covers the baseline 6x7 container array representing a topography that is small in comparison to the size of the city and can be considered negligible. The second section presents the results of the hemisphere case. The hemisphere represents terrain that is affecting the uniform flow through the city and is roughly the same size as the city. The third section presents the results from the large wall case. The wall represents terrain that also affects the uniform flow through the city but on a much larger scale. The size of the wall is a couple orders of magnitude larger than the city.

The constant Z contour slices of the velocity differences provide a qualitative view of the performance of the data assimilation technique in QUIC-URB. These contour plots provide a comparison to the FLUENT solutions and the previous

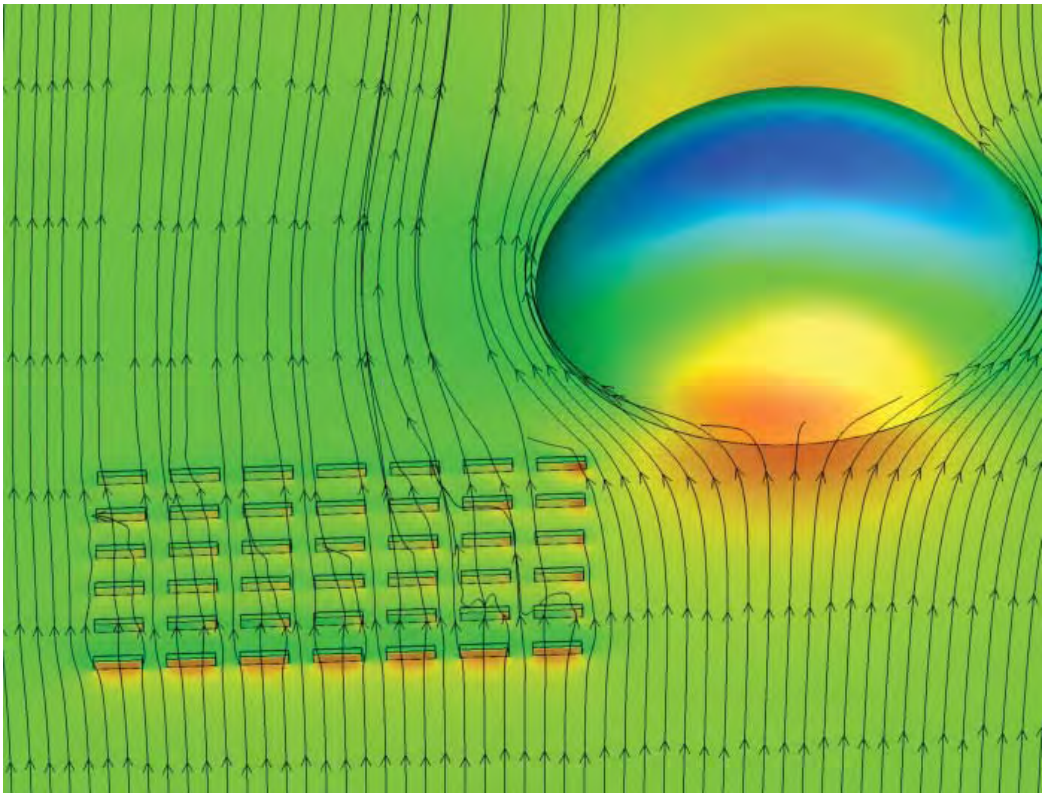

Figure 5.2. Streamlines and surface pressure contours for the 6x7 array and hemisphere from the FLUENT solution.

version of QUIC-URB before the Barnes data assimilation technique was added. The single profile QUIC-URB solution is equivalent to the previous QUIC-URB version since that was its limit for input. Each subplot shows the difference between the FLUENT velocity field and the QUIC-URB velocity field. Specifically, the difference is

$$\vec{U}_{true\_error} = \vec{U}_{FLUENT} - \vec{U}_{QUIC-URB}, \tag{5.1}$$

where the true error wind vector ($\vec{U}_{true\_error}$) is broken up into the three cartesian coordinate components U, V, and W representing the X, Y and Z directional velocities, respectively. To call this true error is to assume that the CFD solution is 100% correct. While this is definitely not the case, it will be treated as though

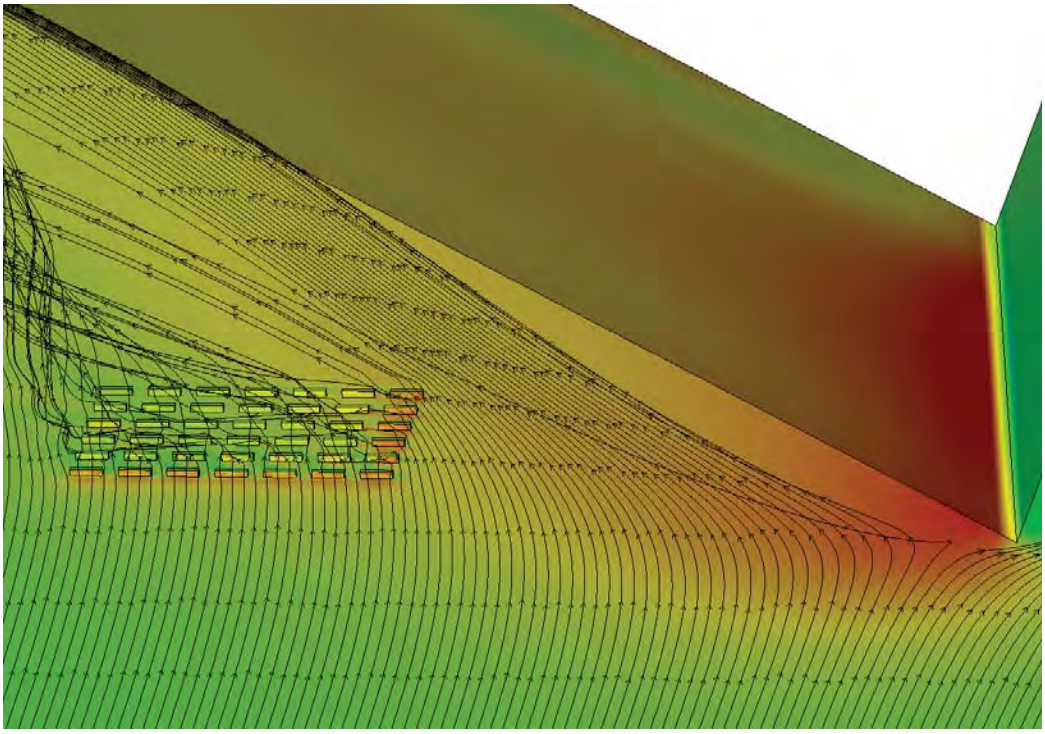

Figure 5.3. Streamlines and surface pressure contours for the 6x7 array and wall from the FLUENT solution.

it is for this analysis since that is what we are trying to match. In reality this is simply just a comparison of two codes. Subsections 5.1.1 to 5.1.3 contain tables of figures of constant Z contour slices at a given height for each of the nine profile configurations used. The contour is a U, V, or W component of the $\vec{U}_{true\_error}$. The title of each subplot gives a summary of what is displayed, while the $\oplus$ data marker in each subplot is the location of the data extracted from the FLUENT solution.

The colormap used for the contours is meant to display the errors quickly and efficiently. Any shade of blue indicates a positive $\vec{U}_{true\_error}$, while shades of red indicate negative $\vec{U}_{true\_error}$. The darker the shade of red or blue the larger the magnitude of negative or positive $\vec{U}_{true\_error}$, respectively. Therefore, large amounts of white and light shades of red and blue in a picture indicates a smaller

amount of error. For contour slices below 2.5 meters the buildings are shown in black.

5.1.1 Baseline 6x7 MUST array

The results in this section display the QUIC-URB to FLUENT velocity difference comparisons for the baseline 6x7 container array. Figures 5.4 - 5.7 show the $\vec{U}_{true\_error}$ for each of the QUIC-URB sensor configurations at heights of 0.25, 1.25, 2.75, and 5.25 meters above the ground for the streamwise (V) component of velocity. These heights were chosen to compare the results near the ground, half way up the containers, just above the top of the containers and at two times the height of the containers. Comparisons at all the heights and the other two velocity components were produced but were left out of this report for brevity. The QUIC-URB solutions that ingested more than four profiles shown in these figures show that the internal urban canopy profiles have a radius of influence on the flow field that extends beyond the bounds of the local measured flow. The large blue areas outside of the array show a velocity deficit region that should not be present.

These figures also show that there is a larger wake behind the entire urban canopy that is not being resolved with QUIC-URB's empirical parameterizations and placing a profile within this wake does not quite fix this problem. The contour plot with four profiles shows that the profile within the wake of the array also has a radius of influence on the flow that is larger than the urban wake. This is seen by the blue regions on the sides of the downwind urban wake.

Figure 5.8 shows the spanwise (U) velocity difference contour half way up the sides of the containers. Most of the differences seen with the spanwise component of velocity happen in the urban array wake and within the urban canopy.

Figure 5.9 shows the vertical (W) velocity component on the contour plots. For this simple solution with no topography present ignoring the vertical velocity component in the data assimilation technique appears to have little effect. The majority of the differences seen are within the canopy. These differences are likely due to the empirical parameterizations which could be improved with a more

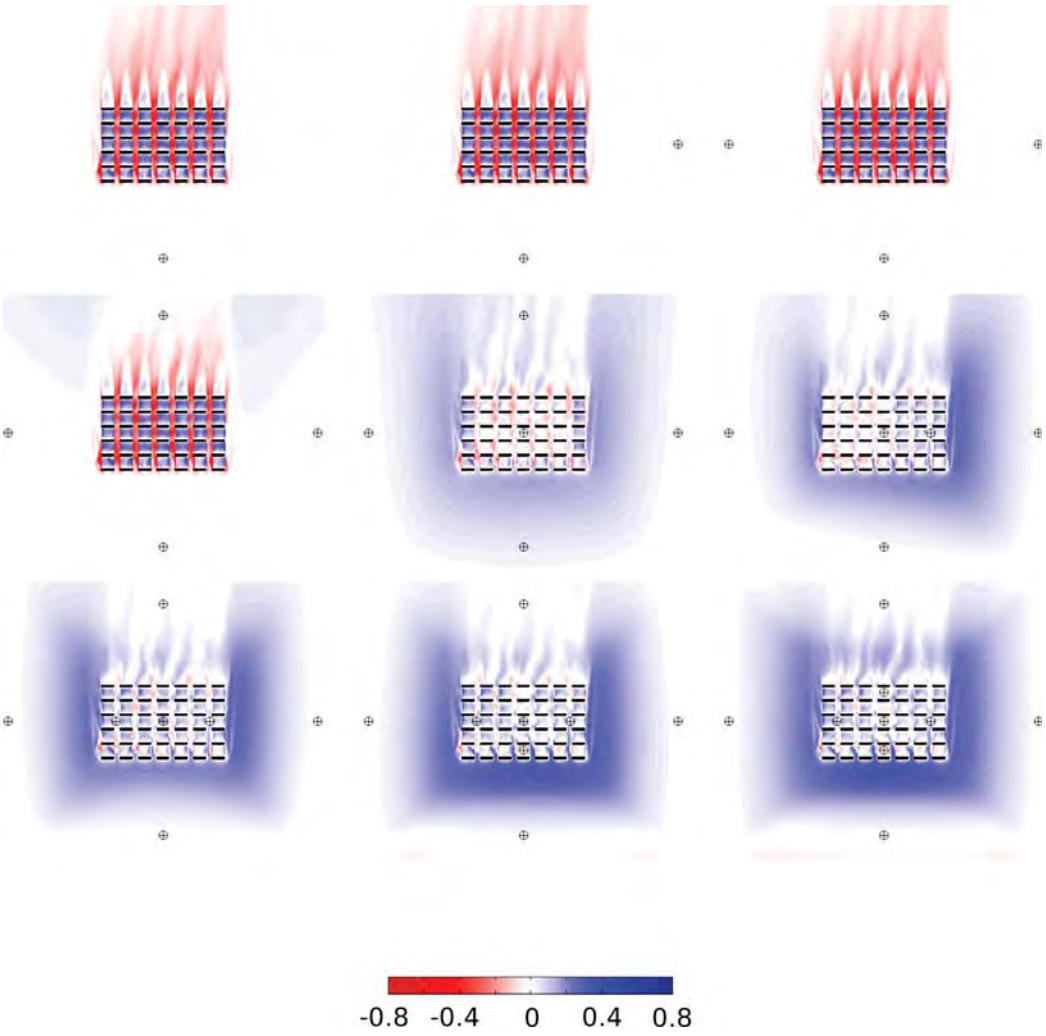

Figure 5.4. Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 0.25 meters for the baseline 6x7 array. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

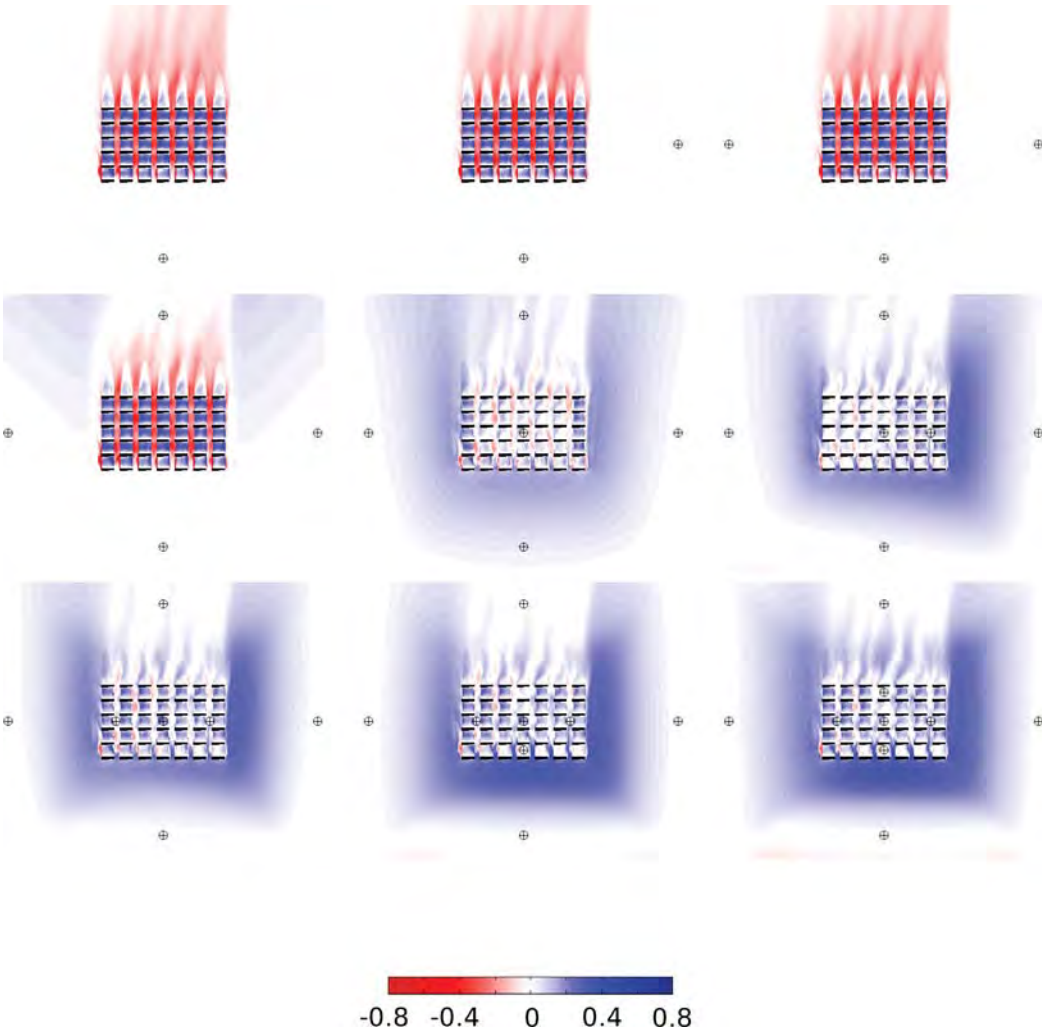

Figure 5.5. Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 1.25 meters for the baseline 6x7 array. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

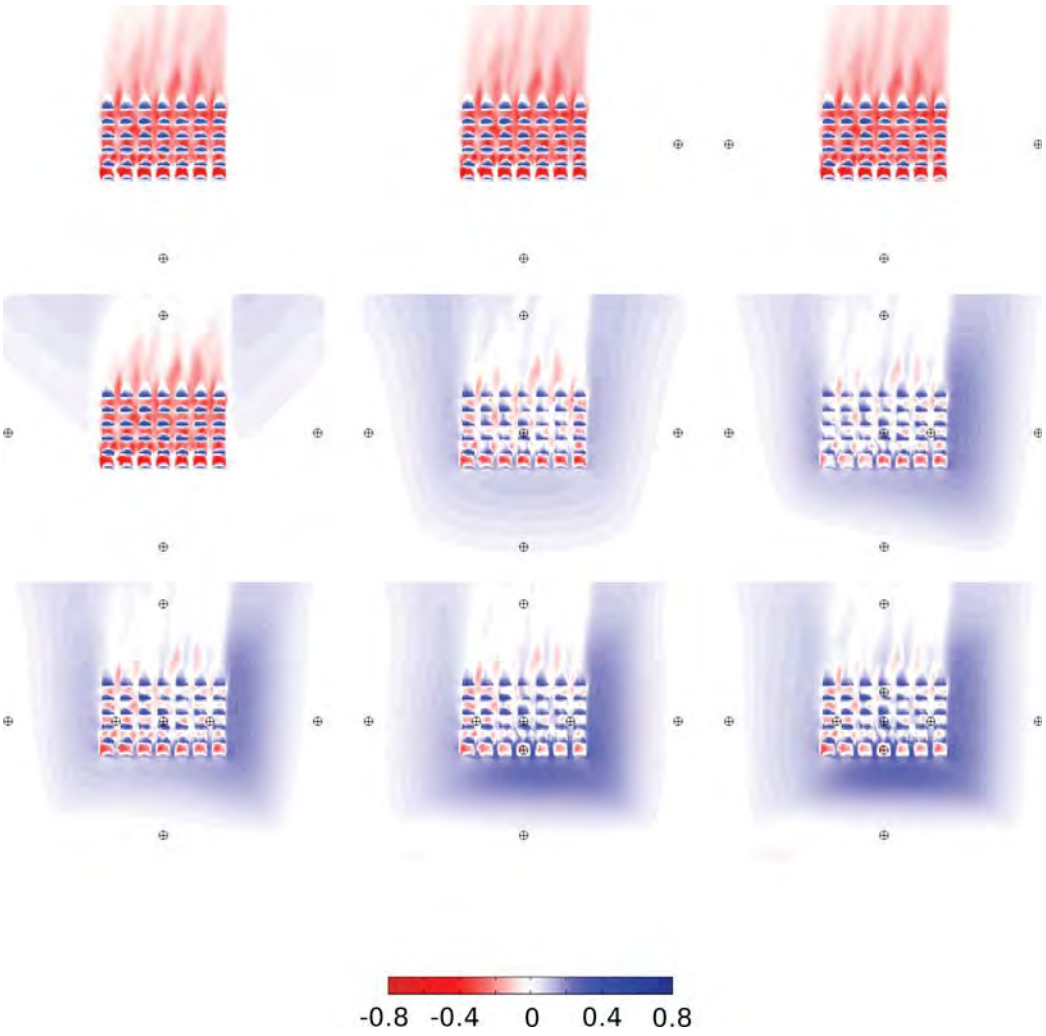

Figure 5.6. Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 2.75 meters for the baseline 6x7 array. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

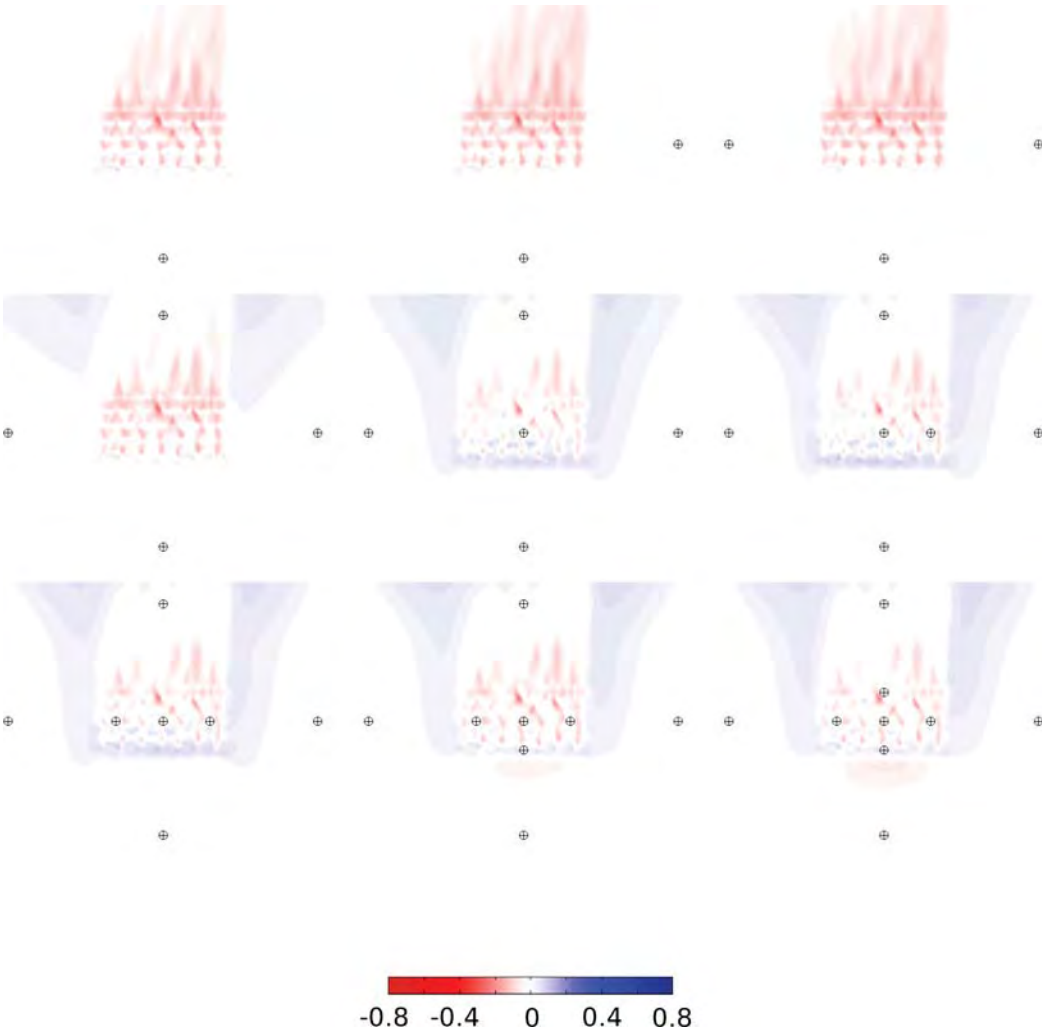

Figure 5.7. Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 5.25 meters for the baseline 6x7 array. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

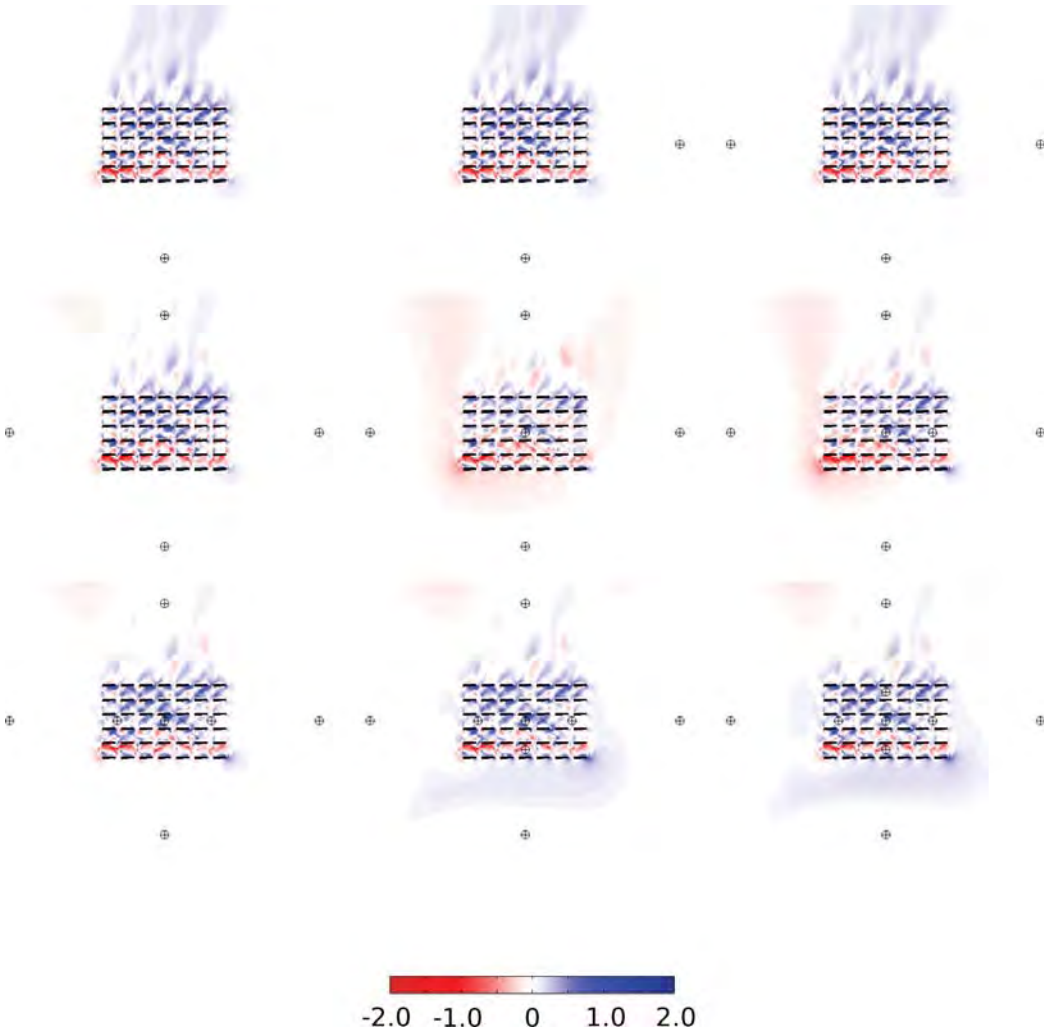

Figure 5.8. Contour slice of the difference in the spanwise (U) velocity of the FLUENT and QUIC-URB solutions at a height of 1.25 meters for the baseline 6x7 array. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

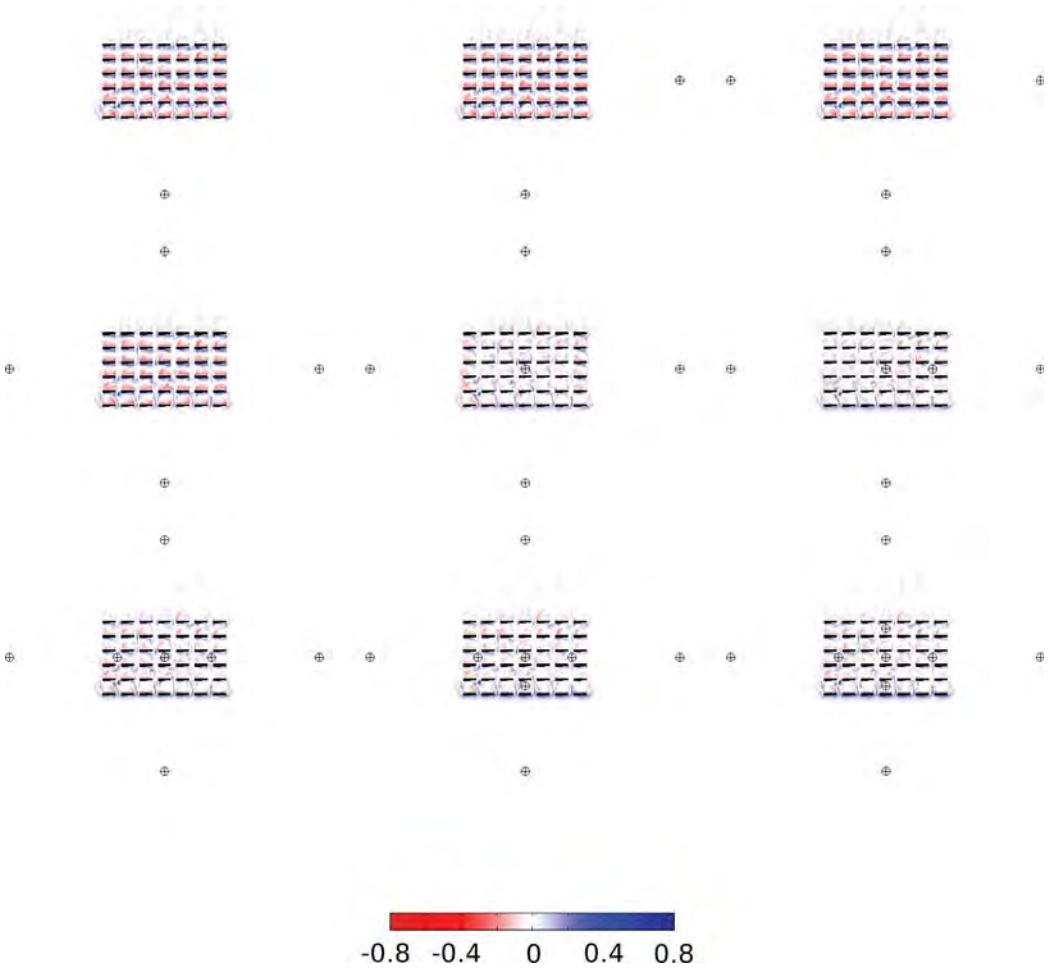

Figure 5.9. Contour slice of the difference in the vertical (W) velocity of the FLUENT and QUIC-URB solutions at a height of 1.25 meters for the baseline 6x7 array. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

intelligent data assimilation technique that recognized the local building flow.

Table 5.1 displays the NRMSE calculations for each 1.5 meter vertical section for all nine profile arrangements. The NRMSE was calculated for each U, V, and W velocity component as well as the velocity magnitude.

The streamwise component and velocity magnitude of the NRMSE in this table show a significant difference between the single profile solution and the two and three profile solutions. The single profile NRMSE is about 1 OOM larger. These solutions (and the NRMSE) should be nearly identical for this baseline 6x7 container array. The velocity difference contour plots do not appear to show differences of this magnitude.

Additional NRMSE calculations and velocity difference plots were created to show the differences between the single profile and 2-3 profile QUIC-URB solutions. The single sensor QUIC-URB solution was used as the baseline data and the other multiprofile QUIC-URB solutions were subtracted from this. Equation 5.2 shows the change in variables from equation 5.1.

$$\vec{U}_{diff} = \vec{U}_{QUICURB_{1prof}} - \vec{U}_{QUICURB_{multiprof}}. \tag{5.2}$$

Figure 5.10 shows the streamwise velocity component of the $\vec{U}_{diff}$ at 1.25 meters high. Again, the solutions with one, two and three profiles should be nearly identical for this baseline case, but this figure highlight the small differences within the canopy. These differences would have to be fairly significant to create such a large difference in the NRMSE calculation. This figure is also shows the differences between the last version of QUIC-URB and this Barnes data assimilation version of QUIC-URB.

Table 5.2 contains the NRMSE calculated for the first three QUIC-URB profiles that should be nearly identical for this baseline case.

For this NRMSE calculation the single profile QUIC-URB solution is the estimator ($x_{1,i}$) and the multiprofile solutions are the estimated parameter ($x_{2,i}$) from equation 4.2.

Table 5.1. Normalized Root Mean Square Error (NRMSE) of the FLUENT and QUIC-URB solutions for the baseline 6x7 container array at four 1.5 meter vertical sections for the spanwise (U) velocity, streamwise (V) velocity, the vertical (W) velocity and the velocity magnitude (Mag). The FLUENT solution is the estimator and the QUIC-URB solution is the estimated parameter in equation 4.2

Vertical Section	**Velocity Comp.**	**Profiles 1**	**Profiles 1,2**	**Profiles 1-3**	**Profiles 1-4**	**Profiles 1-5**	**Profiles 1-6**	**Profiles 1-7**	**Profiles 1-8**	**Profiles 1-9**
0-1.5 m	U	0.0031	0.0023	0.0023	0.0089	0.0060	0.0032	0.0035	0.0047	0.0047
	V	0.0263	0.0029	0.0032	0.0305	0.0190	0.0087	0.0051	0.0243	0.0184
	W	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	Mag	0.0294	0.0053	0.0055	0.0394	0.0251	0.0120	0.0086	0.0291	0.0231
1.5-3.0 m	U	0.0057	0.0043	0.0043	0.0168	0.0089	0.0132	0.0111	0.0074	0.0064
	V	0.0301	0.0043	0.0046	0.0306	0.0184	0.0116	0.0013	0.0156	0.0119
	W	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004
	Mag	0.0361	0.0090	0.0093	0.0477	0.0276	0.0252	0.0127	0.0234	0.0187
3.0-4.5 m	U	0.0080	0.0061	0.0060	0.0221	0.0119	0.0173	0.0148	0.0092	0.0083
	V	0.0389	0.0059	0.0063	0.0334	0.0193	0.0132	0.0090	0.0007	0.0022
	W	0.0008	0.0008	0.0008	0.0008	0.0008	0.0008	0.0008	0.0008	0.0008
	Mag	0.0477	0.0128	0.0131	0.0563	0.0319	0.0312	0.0246	0.0108	0.0113
4.5-6.0 m	U	0.0099	0.0076	0.0075	0.0235	0.0137	0.0181	0.0157	0.0119	0.0107
	V	0.0542	0.0084	0.0089	0.0344	0.0205	0.0148	0.0117	0.0138	0.0125
	W	0.0018	0.0018	0.0018	0.0018	0.0018	0.0018	0.0018	0.0018	0.0018
	Mag	0.0658	0.0177	0.0182	0.0598	0.0360	0.0347	0.0292	0.0275	0.0250

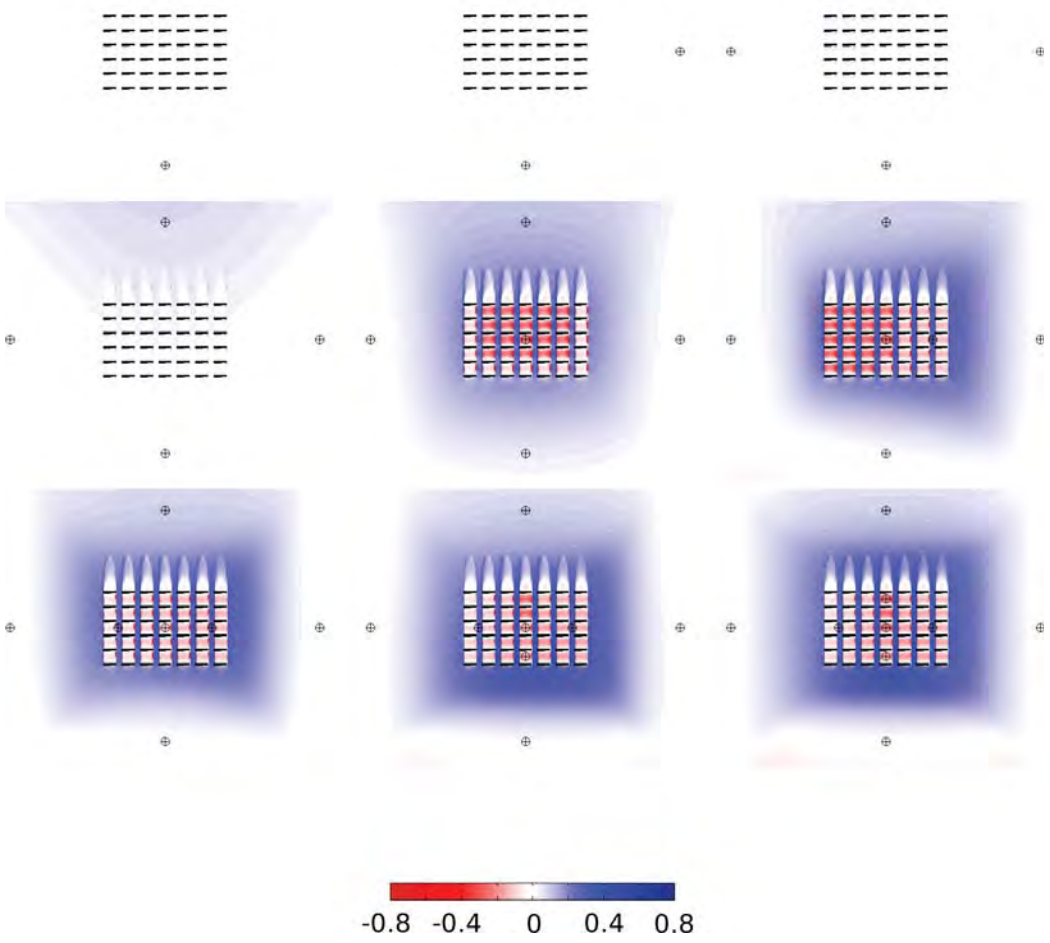

Figure 5.10. Contour slice of the difference in the streamwise (V) velocity of the single sensor QUIC-URB and multisensor QUIC-URB solutions at a height of 1.25 meters for the baseline 6x7 array. Each subplot shows the position of the "sensors" ingested into multisensor QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

Table 5.2. Normalized Root Mean Square Error (NRMSE) of the single sensor QUIC-URB and multisensor QUIC-URB solutions for the baseline 6x7 container array at four 1.5 meter vertical sections for the spanwise (U) velocity, streamwise (V) velocity, the vertical (W) velocity and the velocity magnitude (Mag). The single sensor QUIC-URB solution is the estimator and the multisensor QUIC-URB solution is the estimated parameter in equation 4.2

Vertical Section	**Velocity Comp.**	**Profiles 1**	**Profiles 1,2**	**Profiles 1-3**	**Avg. of (1,2) and (1-3)**
0-1.5 m	U	0.0000	0.0011	0.0011	0.0011
	V	0.0000	0.0200	0.0198	0.0199
	W	0.0000	0.0000	0.0000	0.0000
	Mag	0.0000	0.0211	0.0209	0.0210
1.5-3.0 m	U	0.0000	0.0010	0.0011	0.0011
	V	0.0000	0.0215	0.0212	0.0214
	W	0.0000	0.0000	0.0000	0.0000
	Mag	0.0000	0.0226	0.0223	0.0225
3.0-4.5 m	U	0.0000	0.0010	0.0011	0.0011
	V	0.0000	0.0211	0.0209	0.0210
	W	0.0000	0.0000	0.0000	0.0000
	Mag	0.0000	0.0222	0.0219	0.0221
4.5-6.0 m	U	0.0000	0.0010	0.0010	0.0010
	V	0.0000	0.0203	0.0200	0.0202
	W	0.0000	0.0000	0.0000	0.0000
	Mag	0.0000	0.0212	0.0210	0.0211

The table shows that the error between the single profile and the two and three profile solutions are roughly the same size as the difference between the NRMSE in Table 5.1. The column that shows the comparison of the single profile solution with itself was calculated to validation that the NRMSE is being calculated correctly. This column should always be zero.

The error appears to manifest itself during the comparison from the single profile solution to the two or three profile solutions. Since the errors for the two profile and the three profile solutions are very similar it appears that this error occurs between the single and multiprofile solutions. Therefore, to account for this unknown error the average of the single-to-multiprofile NRMSE will be added to the FLUENT-to-QUIC-URB NRMSE as a rough correction for the purpose of

this analysis. The reason for this error should be the focus of another analysis. The calculated average NRMSE is found in Table 5.2.

Table 5.3 shows the corrected values of the NRMSE for the baseline case. These values better reflect the differences associated with using the data assimilation technique over a single profile case and agree well with the difference contour plots.

This table reveals that the spanwise velocity NRMSE for the single profile case was the lowest for the 0-1.5 meter section. The two and three profile solutions are a close second. The two profile case had the lowest streamwise NRMSE at this height with the three profile solution a close second, while the seven profile case was third lowest.

At this height and every other height for all the other geometries, the vertical (W) NRMSE is the same for every profile solution. The magnitude of the velocity vector NRMSE sums up the results quite nice. At this height profile two and three solutions had the lowest and pretty much the same NRMSE, while the single profile solution was slightly higher. Somewhat unexpectedly, the seven profile solution had very nearly the same NRMSE as the single profile case.

Within the 1.5 to 3.0 meter section the two and three profile solutions have the lowest NRMSE, however the seven profile solution now has a lower NRMSE than the single solution. The 3.0 to 4.5 meter section showed that the eight profile case had the lowest NRMSE, with the nine profile case coming in second and the two and three profile cases were a close third and forth. The 4.5 to 6.0 meter section showed that the nine profile case had the lowest NRMSE by a significant amount. The two and three profile cases were once again very similar and were the next set of lowest NRMSE for the velocity magnitude.

Table 5.3. Corrected Normalized Root Mean Square Error (NRMSE) of the FLUENT and QUIC-URB solutions for the baseline 6x7 container array at four 1.5 meter vertical sections for the spanwise (U) velocity, streamwise (V) velocity, the vertical (W) velocity and the velocity magnitude (Mag). The FLUENT solution is the estimator and the QUIC-URB solution is the estimated parameter in equation 4.2

Vertical Section	**Velocity Comp.**	**Profiles 1**	**Profiles 1,2**	**Profiles 1-3**	**Profiles 1-4**	**Profiles 1-5**	**Profiles 1-6**	**Profiles 1-7**	**Profiles 1-8**	**Profiles 1-9**
0-1.5 m	U	0.0031	0.0034	0.0034	0.0100	0.0071	0.0043	0.0046	0.0058	0.0058
	V	0.0263	0.0228	0.0231	0.0504	0.0389	0.0286	0.0250	0.0442	0.0383
	W	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001	0.0001
	Mag	0.0294	0.0263	0.0265	0.0604	0.0461	0.0330	0.0296	0.0501	0.0441
1.5-3.0 m	U	0.0057	0.0054	0.0054	0.0179	0.0100	0.0143	0.0122	0.0085	0.0075
	V	0.0301	0.0257	0.0260	0.0520	0.0398	0.0330	0.0227	0.0370	0.0333
	W	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004	0.0004
	Mag	0.0361	0.0315	0.0318	0.0702	0.0501	0.0477	0.0352	0.0459	0.0412
3.0-4.5 m	U	0.0080	0.0072	0.0071	0.0232	0.0130	0.0184	0.0159	0.0103	0.0094
	V	0.0389	0.0269	0.0273	0.0544	0.0403	0.0342	0.0300	0.0217	0.0232
	W	0.0008	0.0008	0.0008	0.0008	0.0008	0.0008	0.0008	0.0008	0.0008
	Mag	0.0477	0.0349	0.0352	0.0784	0.0540	0.0533	0.0467	0.0329	0.0334
4.5-6.0 m	U	0.0099	0.0086	0.0085	0.0245	0.0147	0.0191	0.0167	0.0129	0.0117
	V	0.0542	0.0286	0.0291	0.0546	0.0407	0.0350	0.0319	0.0340	0.0327
	W	0.0018	0.0018	0.0018	0.0018	0.0018	0.0018	0.0018	0.0018	0.0018
	Mag	0.0658	0.0388	0.0393	0.0809	0.0571	0.0558	0.0503	0.0486	0.0211

5.1.2 6x7 MUST array and the hemisphere topography

The results in this section display the QUIC-URB to FLUENT velocity difference comparisons for the 6x7 container array with a hemisphere topography. Figures 5.11 - 5.19 show the $\vec{U}_{true\_error}$ for each of the QUIC-URB sensor configurations at heights of 0.25, 1.25, 2.75, and 5.25 meters above the ground for the spanwise (U), streamwise (V) and vertical (W) components of velocity. Additional spanwise velocity difference contours are shown in this section since this solution has a larger percentage of spanwise flow compared to the baseline case.

It is difficult to see many differences between the different profile solutions. While there are some small differences to be seen it is apparent that there are still large amounts of differences between these flows. However, it does appear that the additional internal canopy sensors have reduced the error within the canopy. This profile layout is not ideal for this topography since it is a sparse layout of sensors and the few that are close enough to the nonuniform flow around the hemisphere don't adequately capture its affects on the flow field. With many additional properly placed profiles this comparison could have been better. However, in a real-world situation analysts do not always get to pick the locations of the meteorological sensors. The efficient placement of sensors and the sensitivity to multiplying the number of sensors used should be the topic of another study as well.

An item to note about these figures, there were significant interpolation problems while interpolating the FLUENT solution to the QUIC-URB grid. The FLUENT solutions have no velocity definition within the hemisphere and the QUIC-URB grid did not model the hemisphere, therefore there should be a velocity that gets interpolated to this location that would be inside the hemisphere. The comparisons of the FLUENT solutions to the QUIC-URB solutions show very large differences within the hemisphere since the velocities outside the hemisphere were linearly interpolated inside the canopy to get a solution. Therefore, the area within the hemisphere should be ignored. This area also added a significant amount of NRMSE to these cases so the magnitude of this error should not be

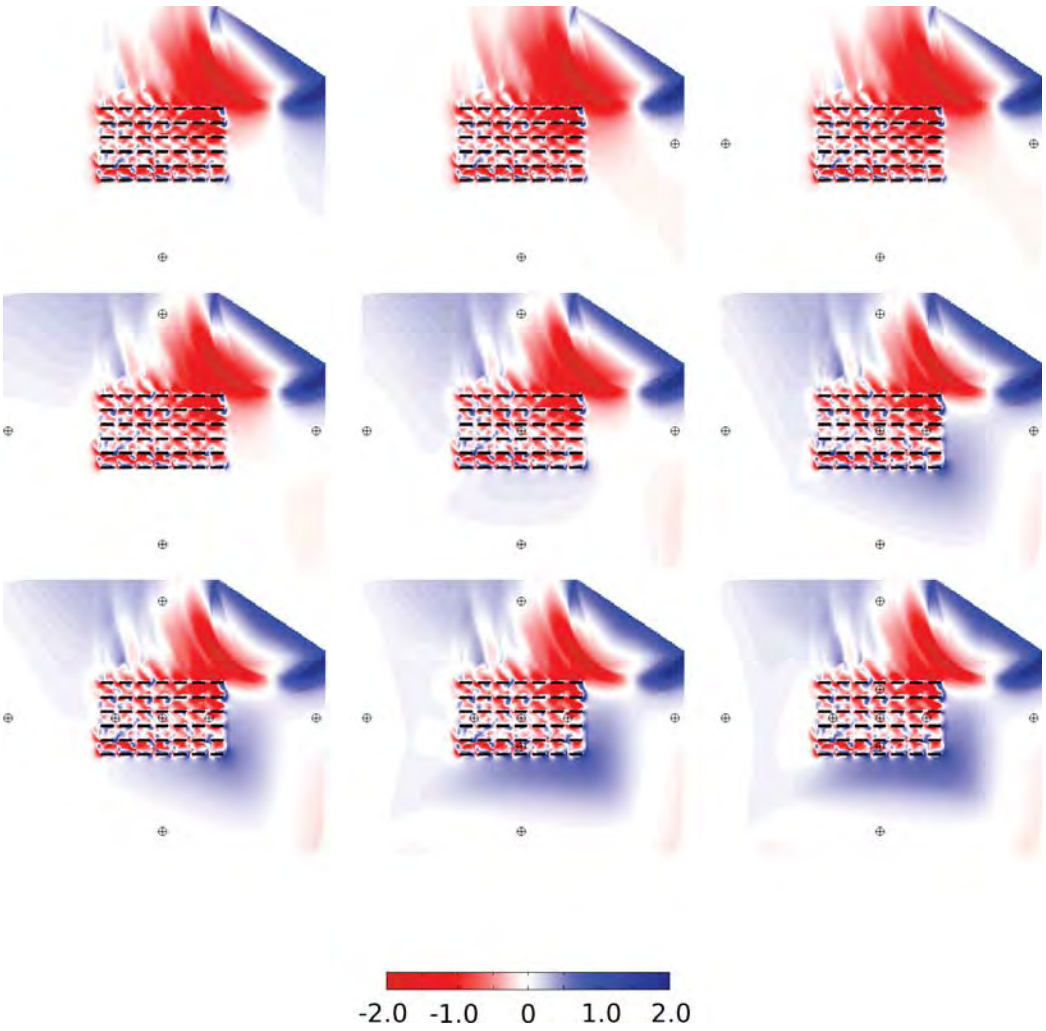

Figure 5.11. Contour slice of the difference in the spanwise (U) velocity of the FLUENT and QUIC-URB solutions at a height of 0.25 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

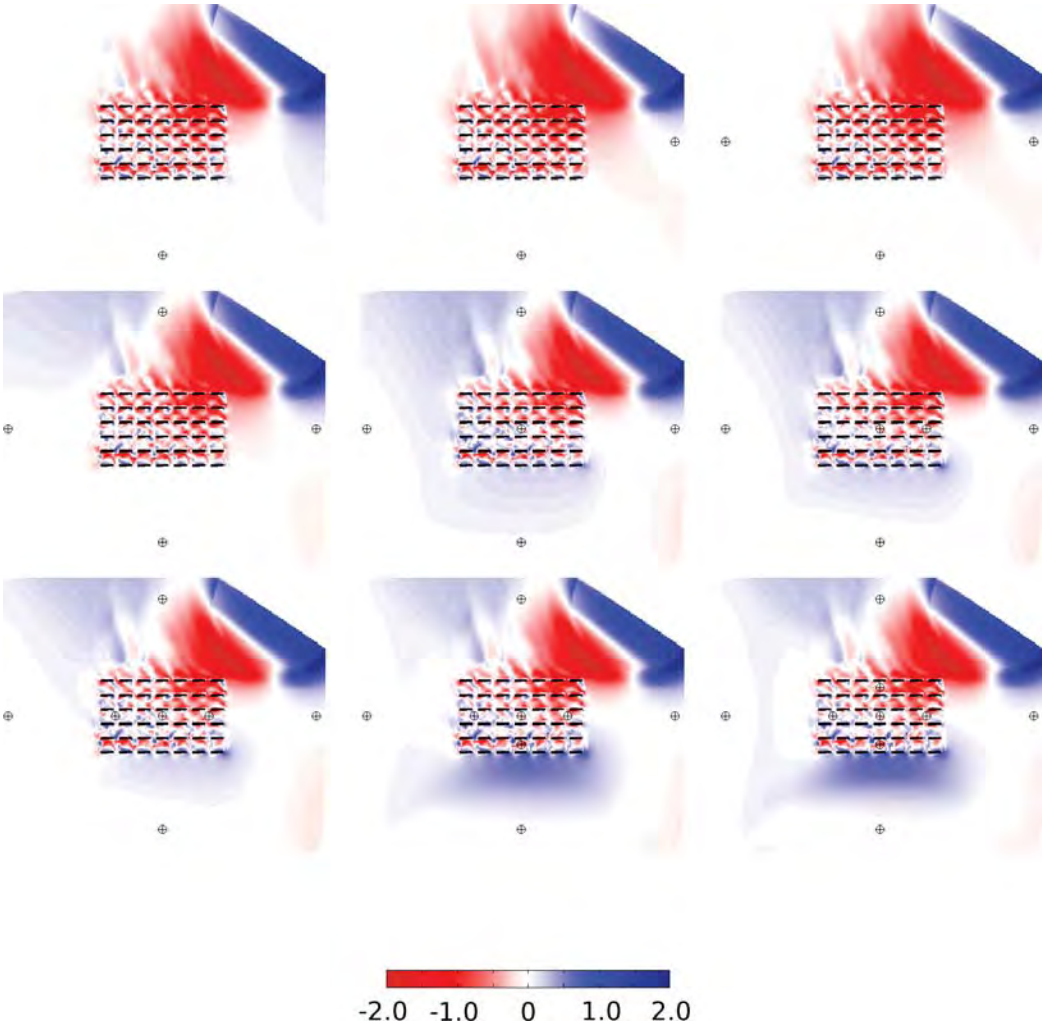

Figure 5.12. Contour slice of the difference in the spanwise (U) velocity of the FLUENT and QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

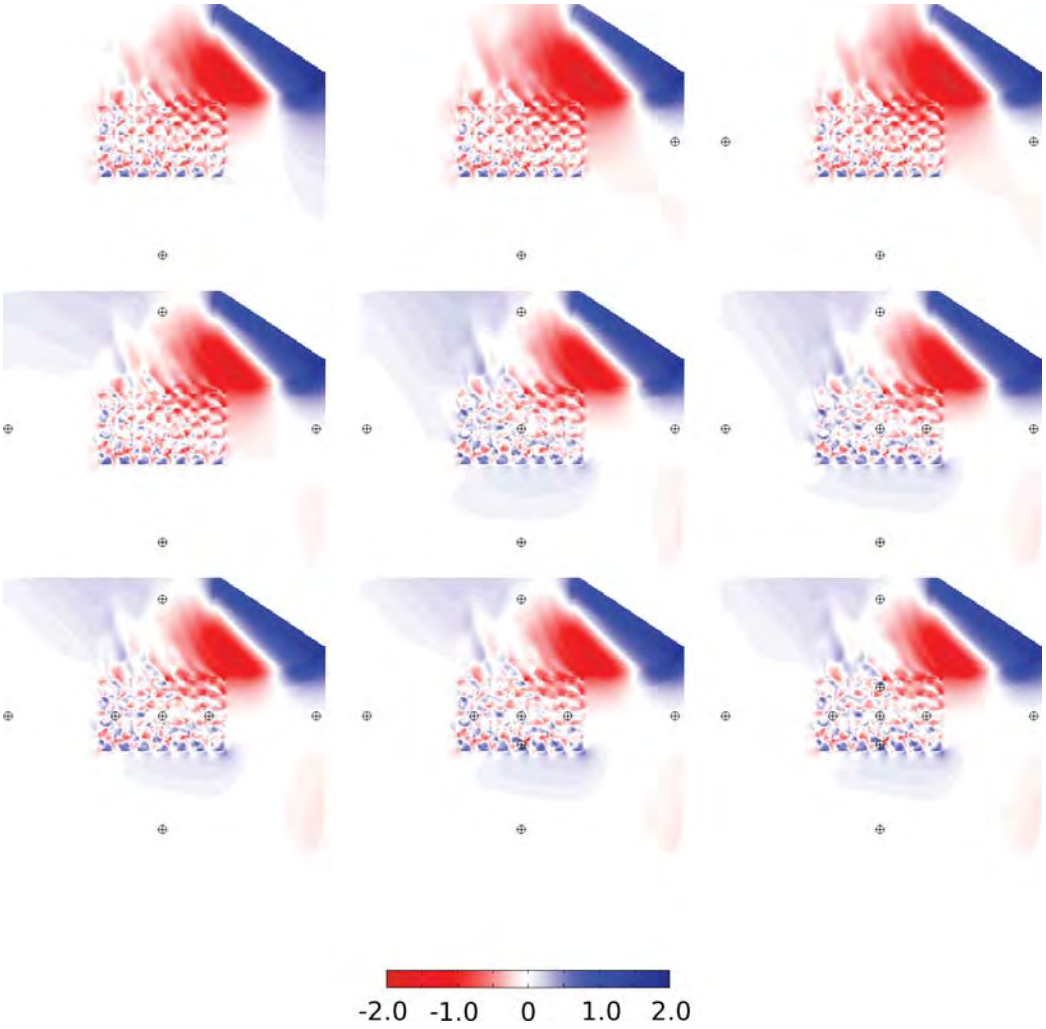

Figure 5.13. Contour slice of the difference in the spanwise (U) velocity of the FLUENT and QUIC-URB solutions at a height of 2.75 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

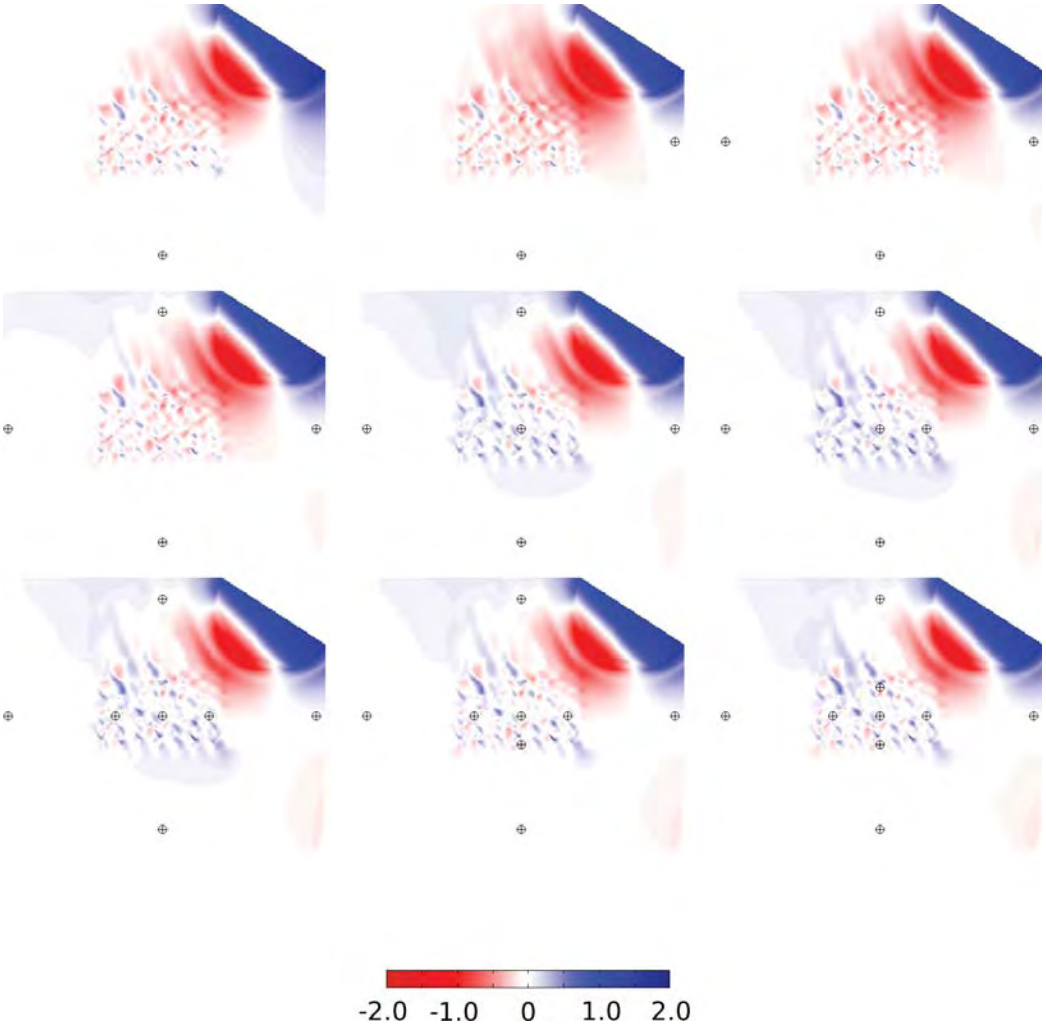

Figure 5.14. Contour slice of the difference in the spanwise (U) velocity of the FLUENT and QUIC-URB solutions at a height of 5.25 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

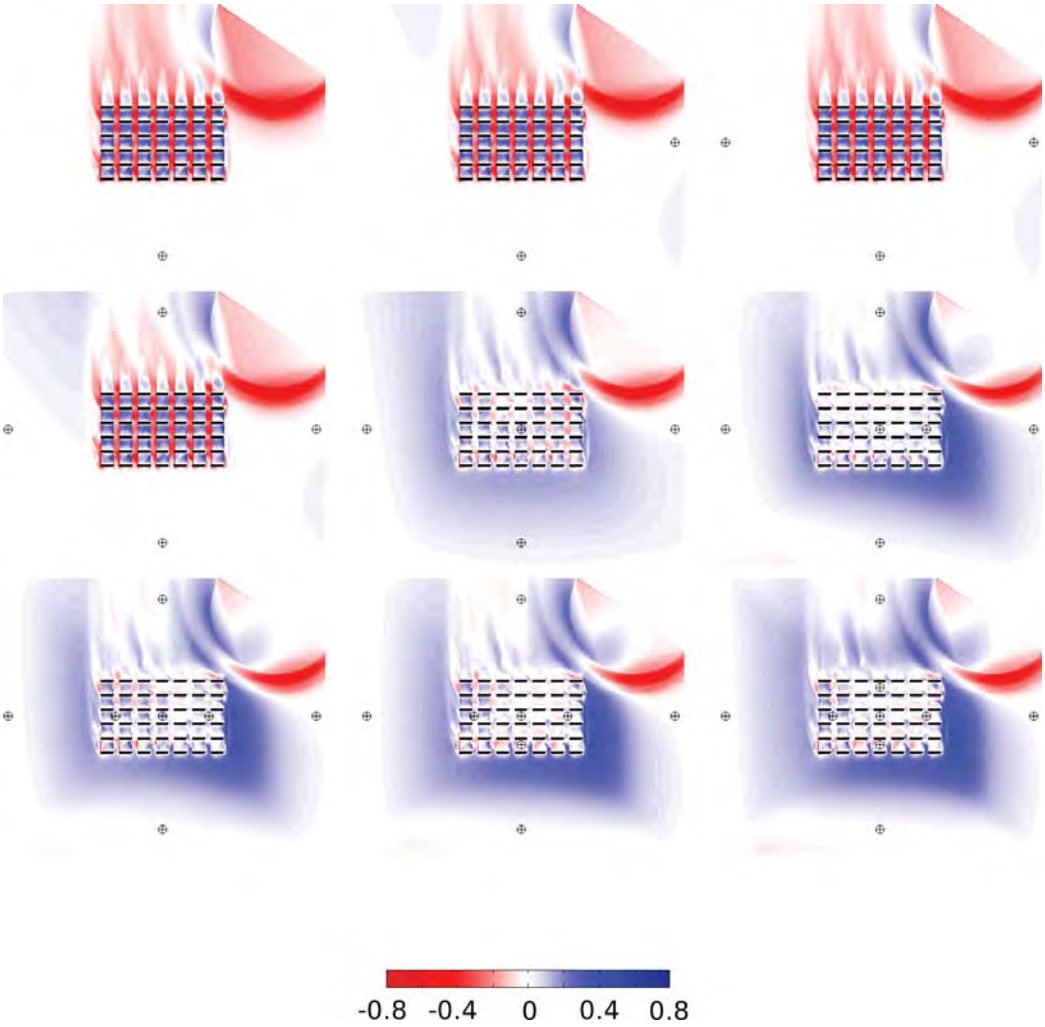

Figure 5.15. Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 0.25 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

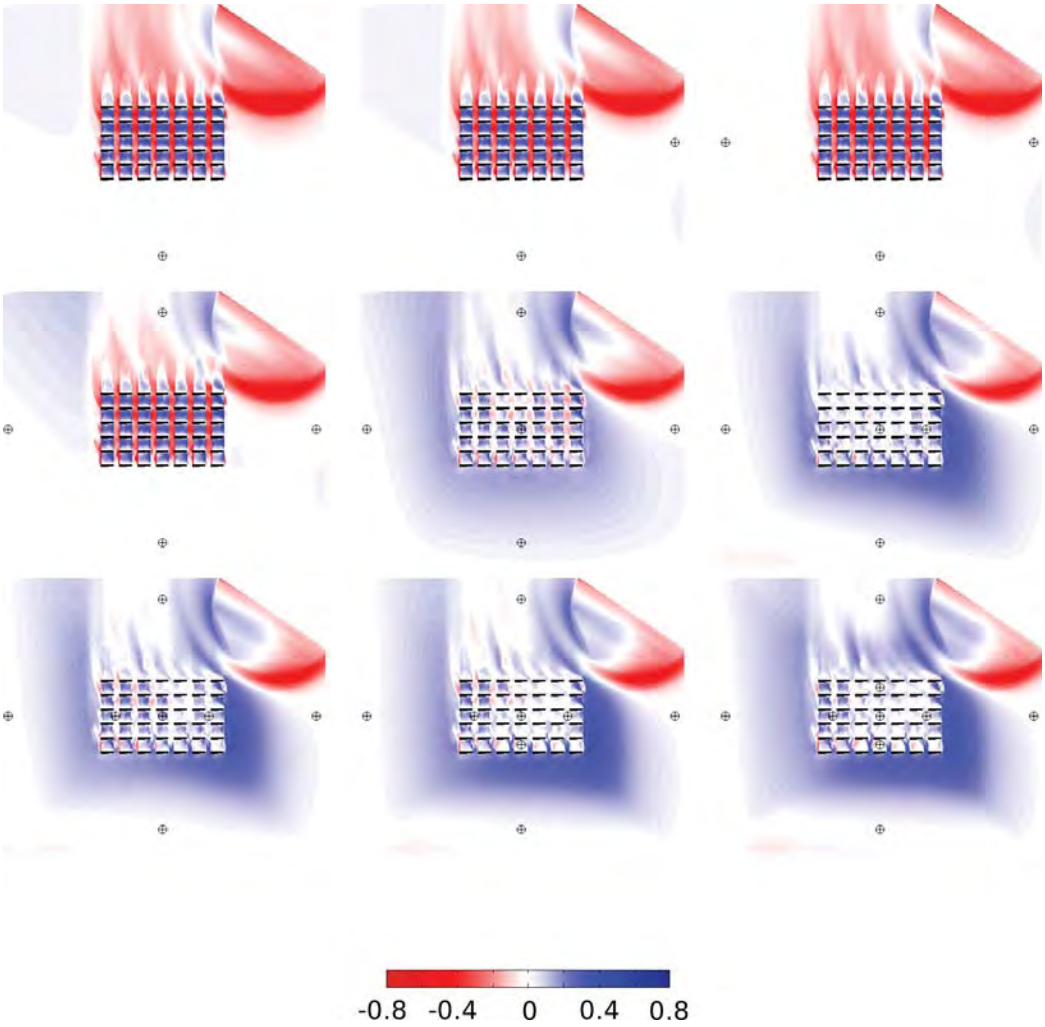

Figure 5.16. Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

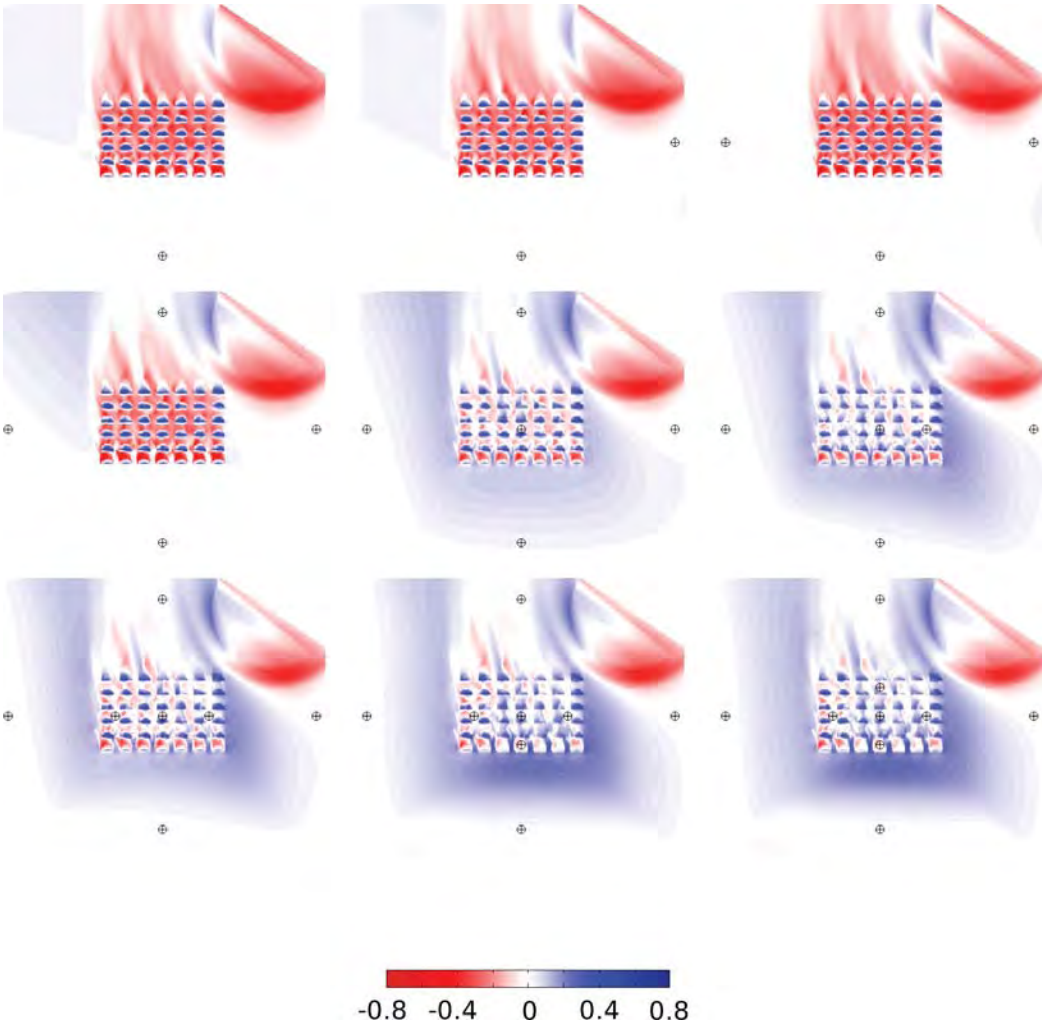

Figure 5.17. Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 2.75 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

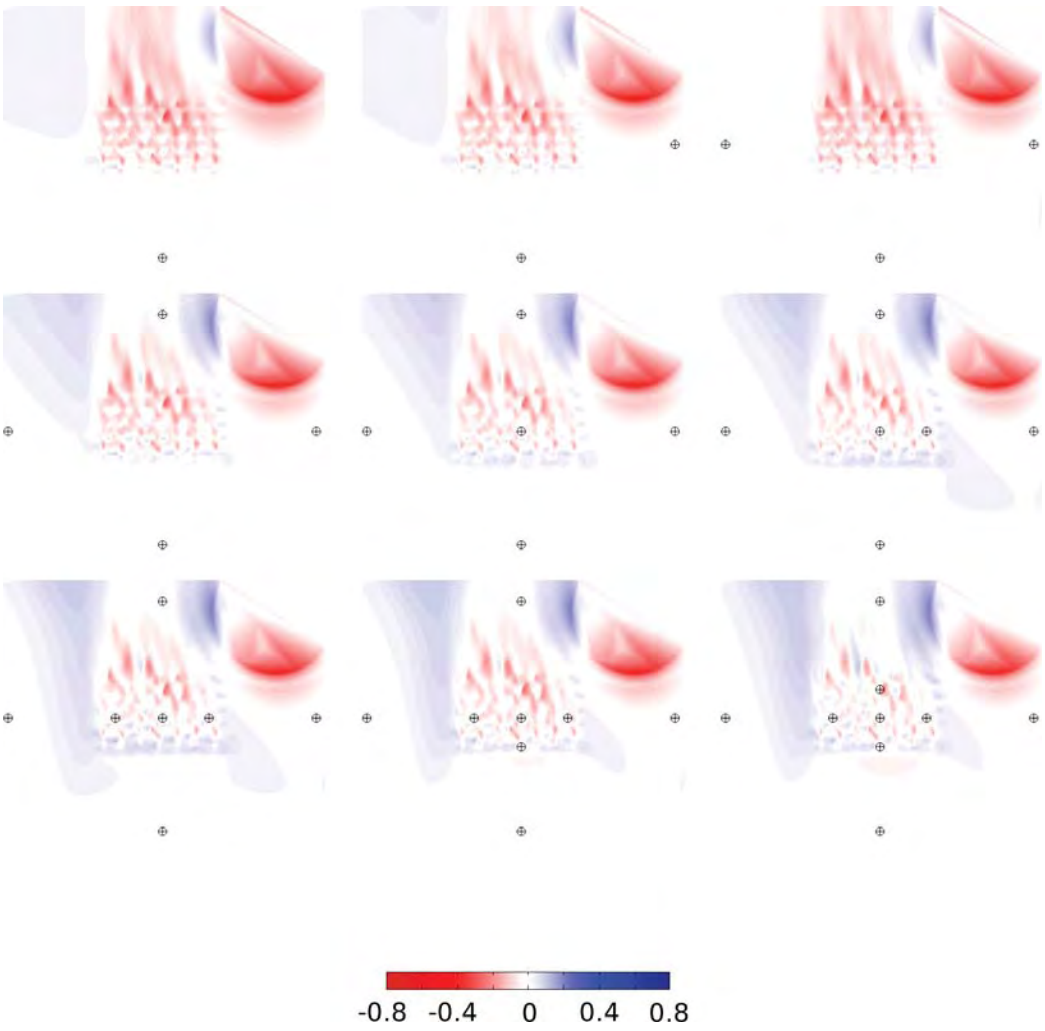

Figure 5.18. Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 5.25 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

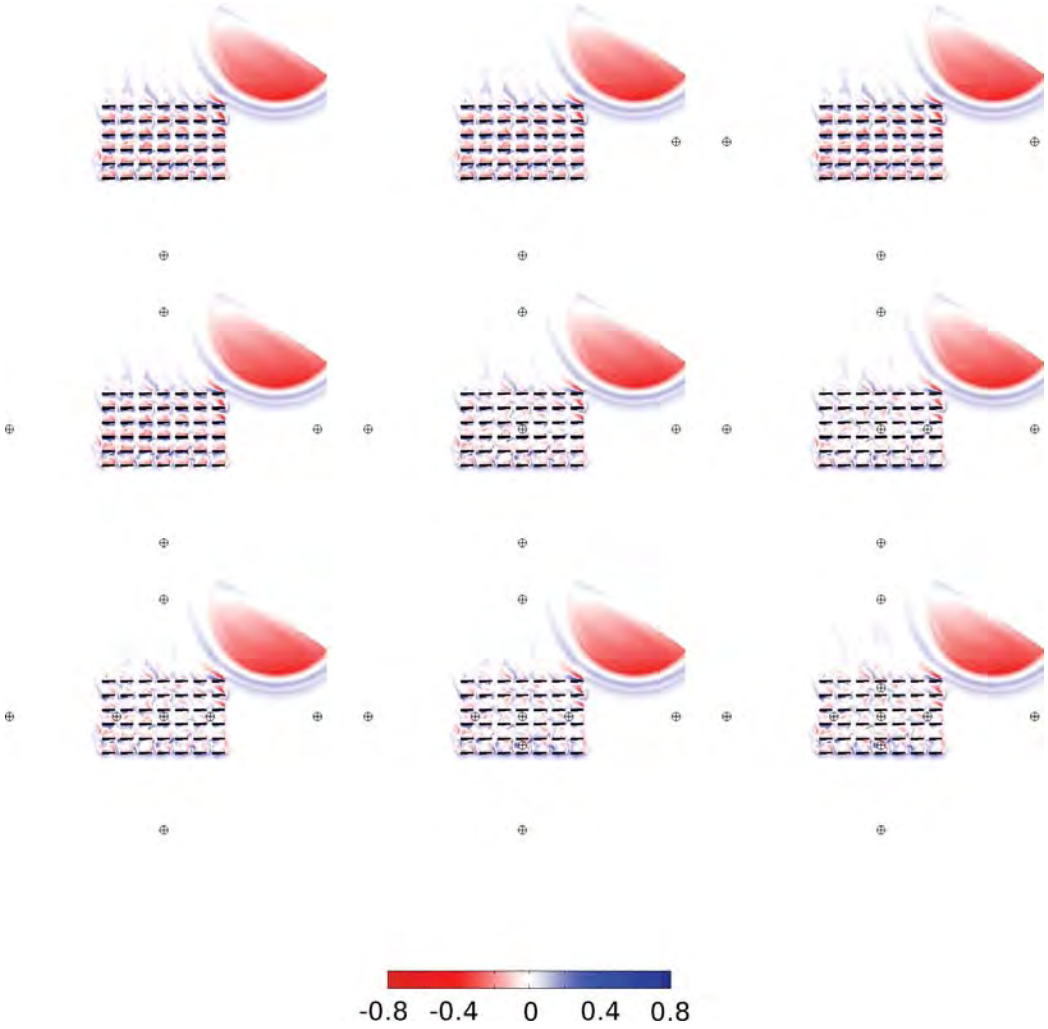

Figure 5.19. Contour slice of the difference in the vertical (W) velocity of the FLUENT and QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

compared directly with other FLUENT topography solutions. Figure 5.19 shows the vertical (W) velocity difference of the $\vec{U}_{true\_error}$ for a reference to the vertical velocity component not being modeled with this current data assimilation technique.

Table 5.4 contains the corrected NRMSE values for these hemisphere solutions. It was also corrected using the average values from Table 5.2.

For the 0 to 1.5 meter section this table shows that the four and five profile solutions had the lowest NRMSE. The two profile solution was third, which was followed by the three and six profile cases which were almost identical. For this geometry it is not expected that the two and three profile solutions will match as closely as they did during the comparison of the baseline case. All of these solutions show a NRMSE more than 1 OOM higher than the baseline solution as well, but once again the NRMSE values for this topography are artificially increased by the interpolation necessary for the internal hemisphere velocities and shouldn't be directly compared.

The 1.5 to 3.0 meter section shows the four and five profile solutions with the lowest and second lowest NRMSE, respectively. Profile two, six and three solutions were close for 3^{rd}, 4^{th} and 5^{th} lowest. The four profile case has the lowest NRMSE for the 3.0 to 4.5 meter section, the five profile case has the 2^{nd} lowest and six and two profile cases had similar NRMSE values for 3^{rd} and 4^{th} lowest. The 4.5 to 6.0 meter section shows similar results with the four and five profile cases showing the lowest NRMSE, however the spread for the rest of the multisensor solutions at this height is much smaller than at the lower heights.

Figures 5.20 to 5.22 show the comparison between the single profile case and the multiprofile cases that used the data assimilation technique. These contours are all at a height of 1.25 meters to show the differences at the middle of the container height. The spanwise and streamwise velocity difference contours show that the sensors are picking up a slight velocity deficit region downwind and to the right of the array compared to the single sensor, however this velocity deficit region should be much bigger than it is. Once again this is more than likely due

Table 5.4. Corrected Normalized Root Mean Square Error (NRMSE) of the FLUENT and QUIC-URB solutions for the 6x7 container array and hemisphere at four 1.5 meter vertical sections for the spanwise (U) velocity, streamwise (V) velocity, the vertical (W) velocity and the velocity magnitude (Mag). The FLUENT solution is the estimator and the QUIC-URB solution is the estimated parameter in equation 4.2

Vertical Section	**Velocity Comp.**	**Profiles 1**	**Profiles 1,2**	**Profiles 1-3**	**Profiles 1-4**	**Profiles 1-5**	**Profiles 1-6**	**Profiles 1-7**	**Profiles 1-8**	**Profiles 1-9**
0-1.5 m	U	0.3157	0.2764	0.2771	0.2928	0.2852	0.2832	0.2811	0.2757	0.2767
	V	0.1420	0.1267	0.1271	0.1061	0.1134	0.1210	0.1336	0.1470	0.1436
	W	0.2029	0.2029	0.2029	0.2029	0.2029	0.2029	0.2029	0.2029	0.2029
	Mag	0.6607	0.6061	0.6070	0.6018	0.6015	0.6071	0.6176	0.6256	0.6232
1.5-3.0 m	U	0.4180	0.3710	0.3717	0.3873	0.3798	0.3782	0.3763	0.3744	0.3750
	V	0.1559	0.1391	0.1395	0.1151	0.1248	0.1323	0.1413	0.1562	0.1527
	W	0.2688	0.2688	0.2688	0.2688	0.2688	0.2688	0.2688	0.2688	0.2688
	Mag	0.8427	0.7789	0.7801	0.7712	0.7734	0.7793	0.7864	0.7994	0.7965
3.0-4.5 m	U	0.4664	0.4139	0.4147	0.4301	0.4228	0.4213	0.4193	0.4195	0.4199
	V	0.1505	0.1276	0.1281	0.1030	0.1134	0.1197	0.1264	0.1372	0.1349
	W	0.2796	0.2796	0.2796	0.2796	0.2796	0.2796	0.2796	0.2796	0.2796
	Mag	0.8965	0.8211	0.8224	0.8127	0.8158	0.8206	0.8253	0.8363	0.8343
4.5-6.0 m	U	0.4531	0.3998	0.4007	0.4138	0.4074	0.4065	0.4048	0.4040	0.4047
	V	0.1048	0.0801	0.0806	0.0649	0.0715	0.0742	0.0762	0.0751	0.0767
	W	0.2091	0.2091	0.2091	0.2091	0.2091	0.2091	0.2091	0.2091	0.2091
	Mag	0.7670	0.6889	0.6903	0.6876	0.6879	0.6897	0.6900	0.6881	0.6903

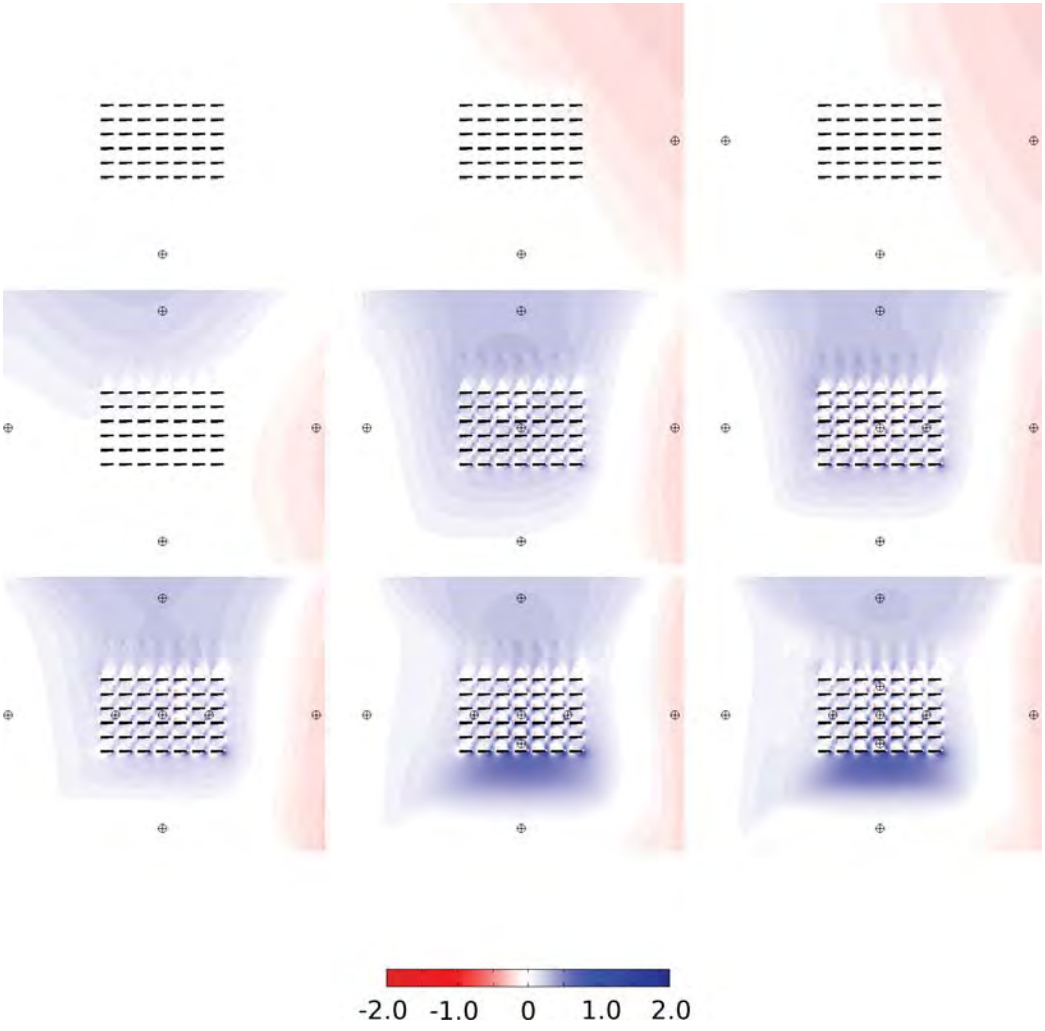

Figure 5.20. Contour slice of the difference in the spanwise (U) velocity of the single sensor QUIC-URB and multisensor QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into multisensor QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

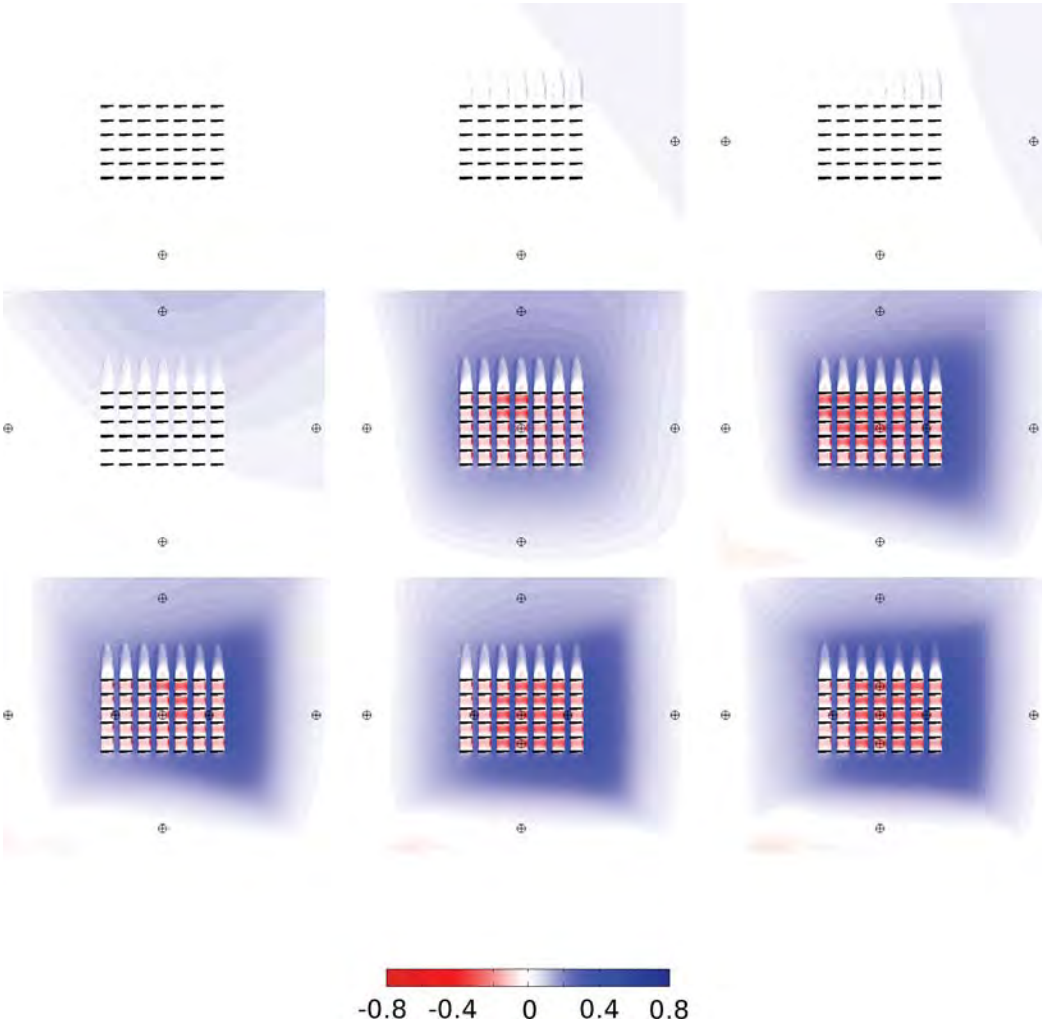

Figure 5.21. Contour slice of the difference in the streamwise (V) velocity of the single sensor QUIC-URB and multisensor QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into multisensor QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

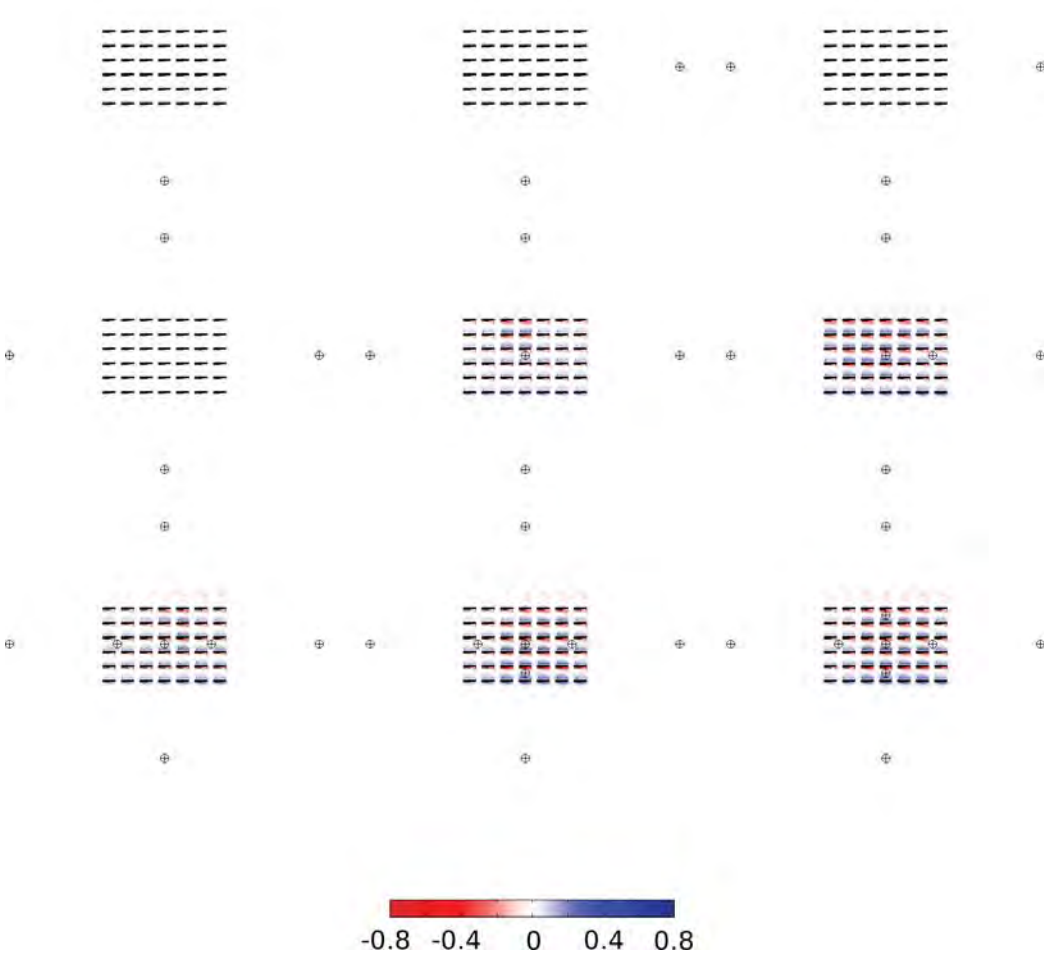

Figure 5.22. Contour slice of the difference in the vertical (W) velocity of the single sensor QUIC-URB and multisensor QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and hemisphere. Each subplot shows the position of the "sensors" ingested into multisensor QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

to the sparse sensor layout. The vertical velocity difference contour shows that there are some differences seen within the array when compared to the single sensor solution.

5.1.3 6x7 MUST array and the large wall topography

The results in this section display the QUIC-URB to FLUENT velocity difference comparisons for the 6x7 container array with the large wall topography. Figures 5.23 - 5.30 show the $\vec{U}_{true\_error}$ for each of the QUIC-URB sensor configurations at heights of 0.25, 1.25, 2.75, and 5.25 meters above the ground for the spanwise (U) and streamwise (V) component of velocity. Once again, additional spanwise velocity difference contours were added since this flow has a very large spanwise component compared to the streamwise component.

This flow has very large inhomogeneous flow conditions that drastically affect the majority of the container array. The streamwise velocity difference plots with four or more profiles show a significant shift of the wakes of the downwind row of the array. As expected, these QUIC-URB wakes match the angle of the FLUENT wake significantly more than the single profile QUIC-URB wakes. The size of the wake from the array is still significantly different, but this is a topic for future work. The velocity difference through the streets also improved significantly with additional internal canopy profiles used, however this adversely affected the external canopy comparison. Once again, these internal canopy profiles have been given a larger radius of influence by the data assimilation technique than they should have due to the sparseness of the sensor layout. Again, this is another topic for future research and arguably would have the biggest impact to this assimilation technique.

Table 5.5 has the corrected NRMSE values for the comparison of the QUIC-URB solution to the FLUENT solution for the wall topography. The average values from baseline Table 5.2 were also used to correct this data.

There is not a single configuration of profiles in this table that has the lowest NRMSE for each of the four vertical sections which is a consistent theme for all three of these topographies.

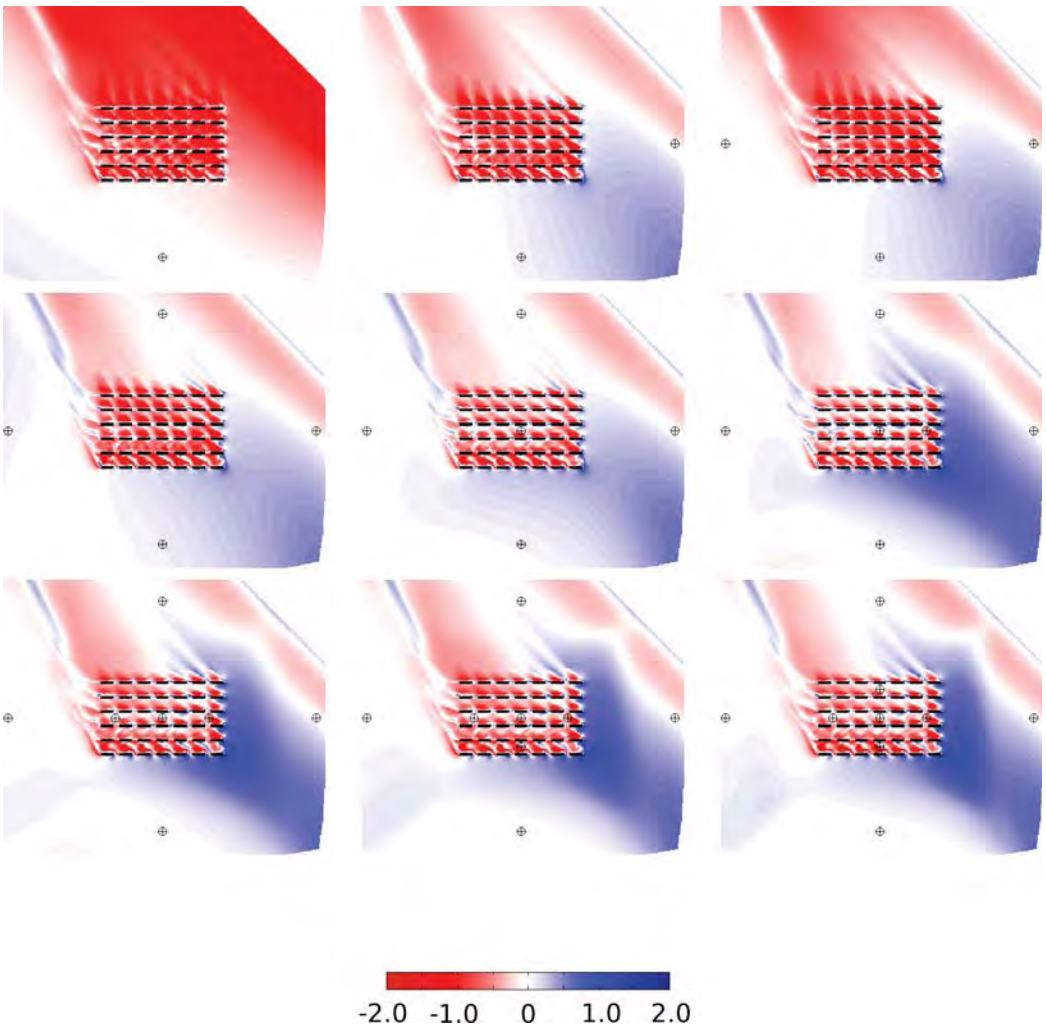

Figure 5.23. Contour slice of the difference in the spanwise (U) velocity of the FLUENT and QUIC-URB solutions at a height of 0.25 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

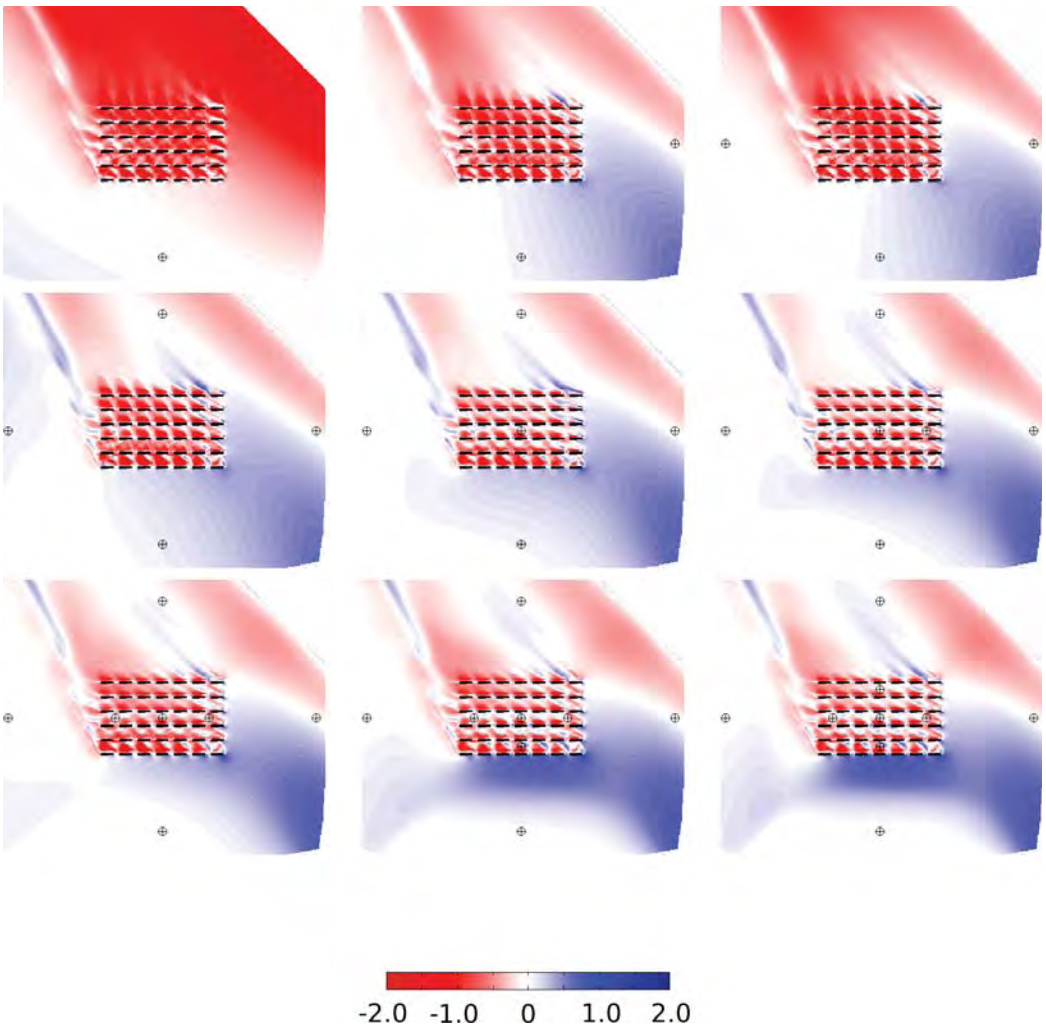

Figure 5.24. Contour slice of the difference in the spanwise (U) velocity of the FLUENT and QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

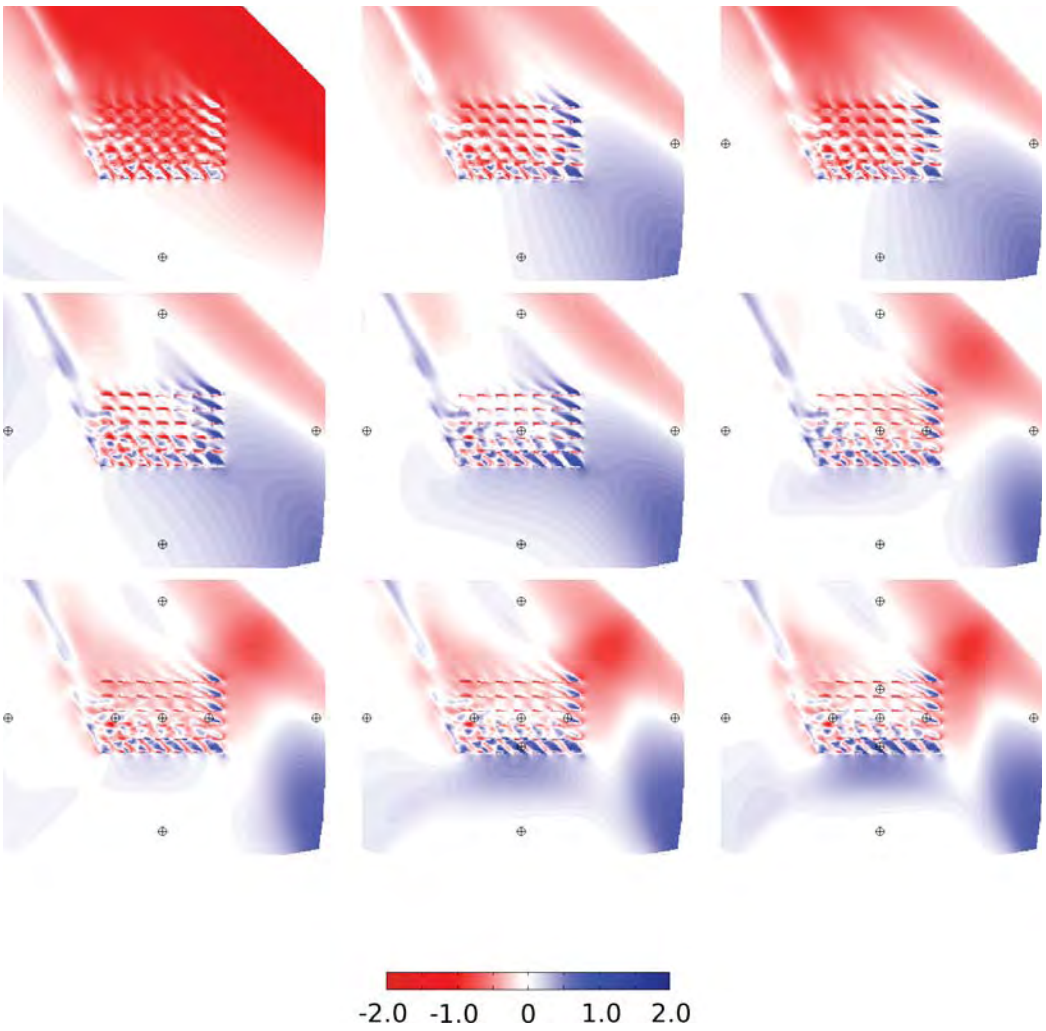

Figure 5.25. Contour slice of the difference in the spanwise (U) velocity of the FLUENT and QUIC-URB solutions at a height of 2.75 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

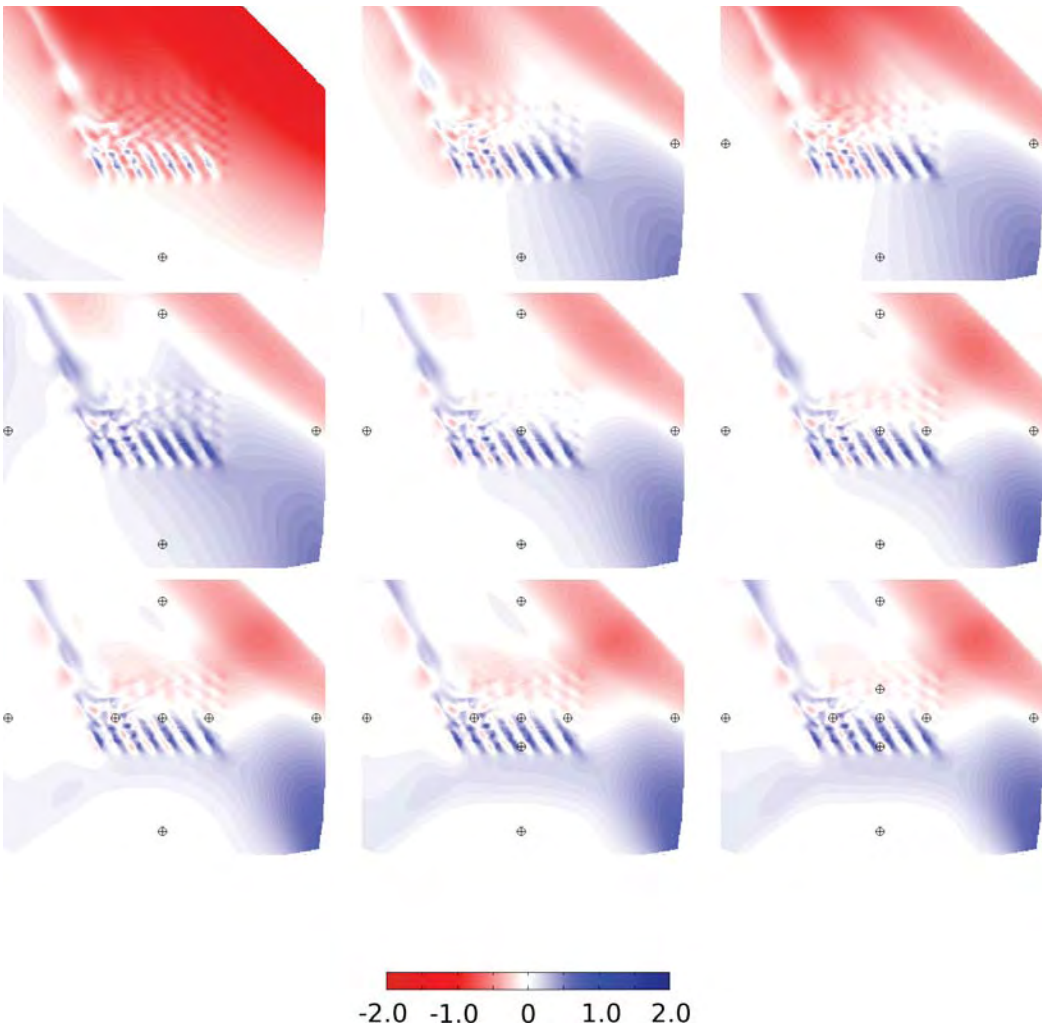

Figure 5.26. Contour slice of the difference in the spanwise (U) velocity of the FLUENT and QUIC-URB solutions at a height of 5.25 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

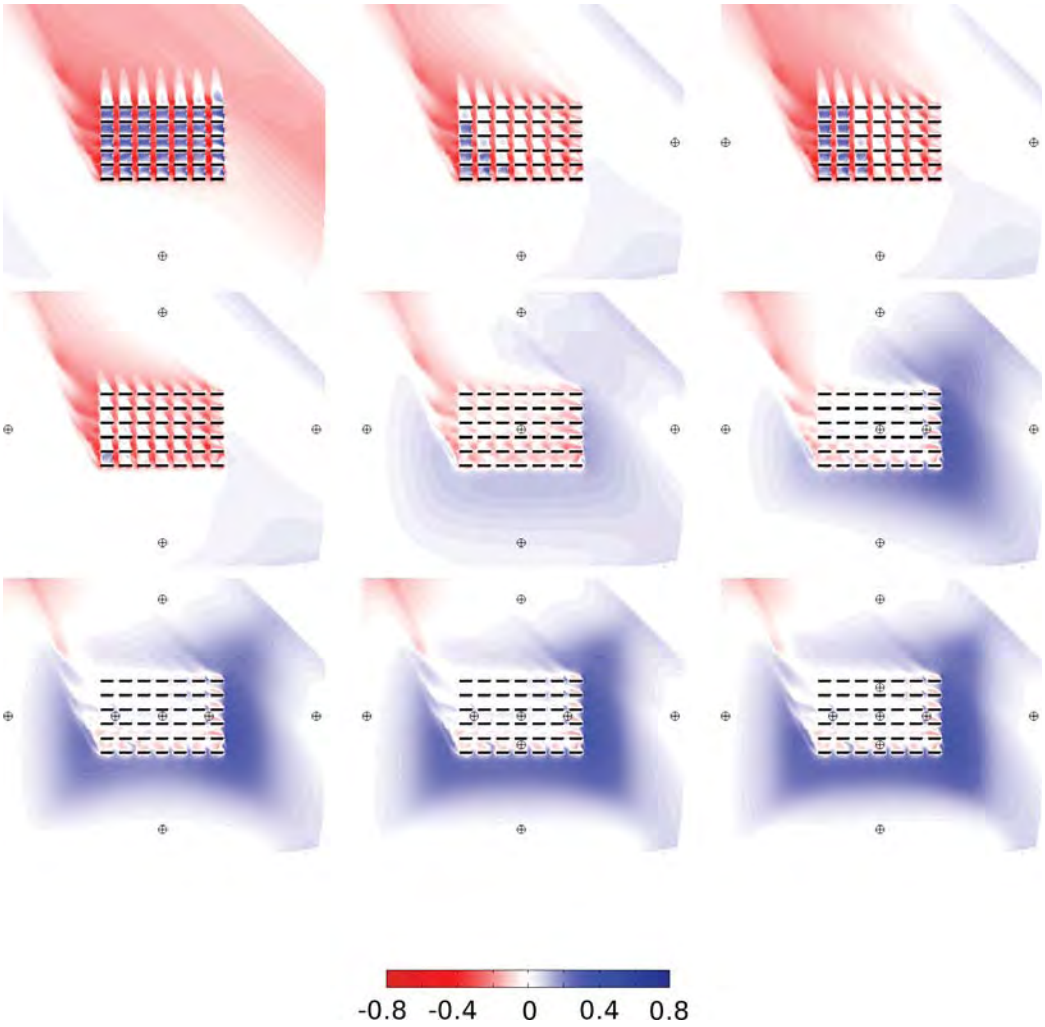

Figure 5.27. Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 0.25 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

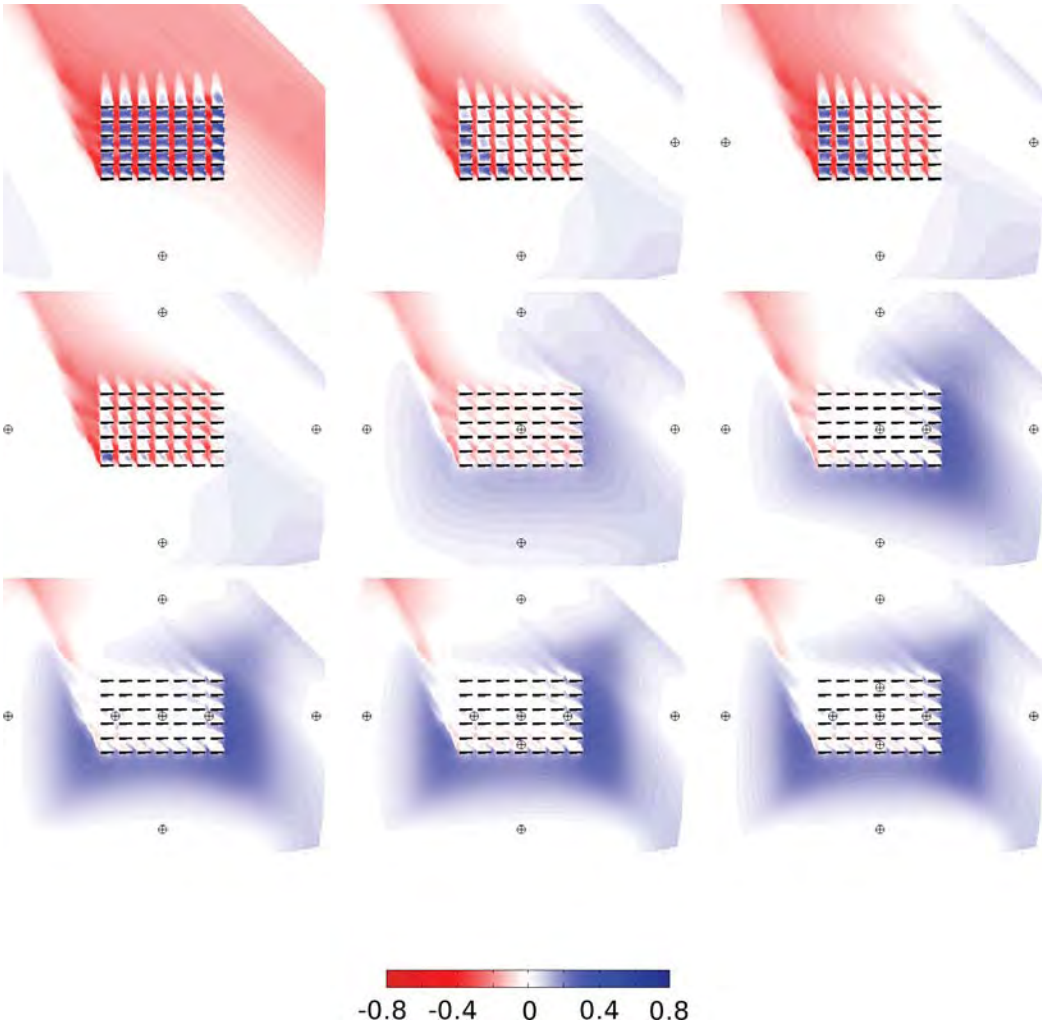

Figure 5.28. Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

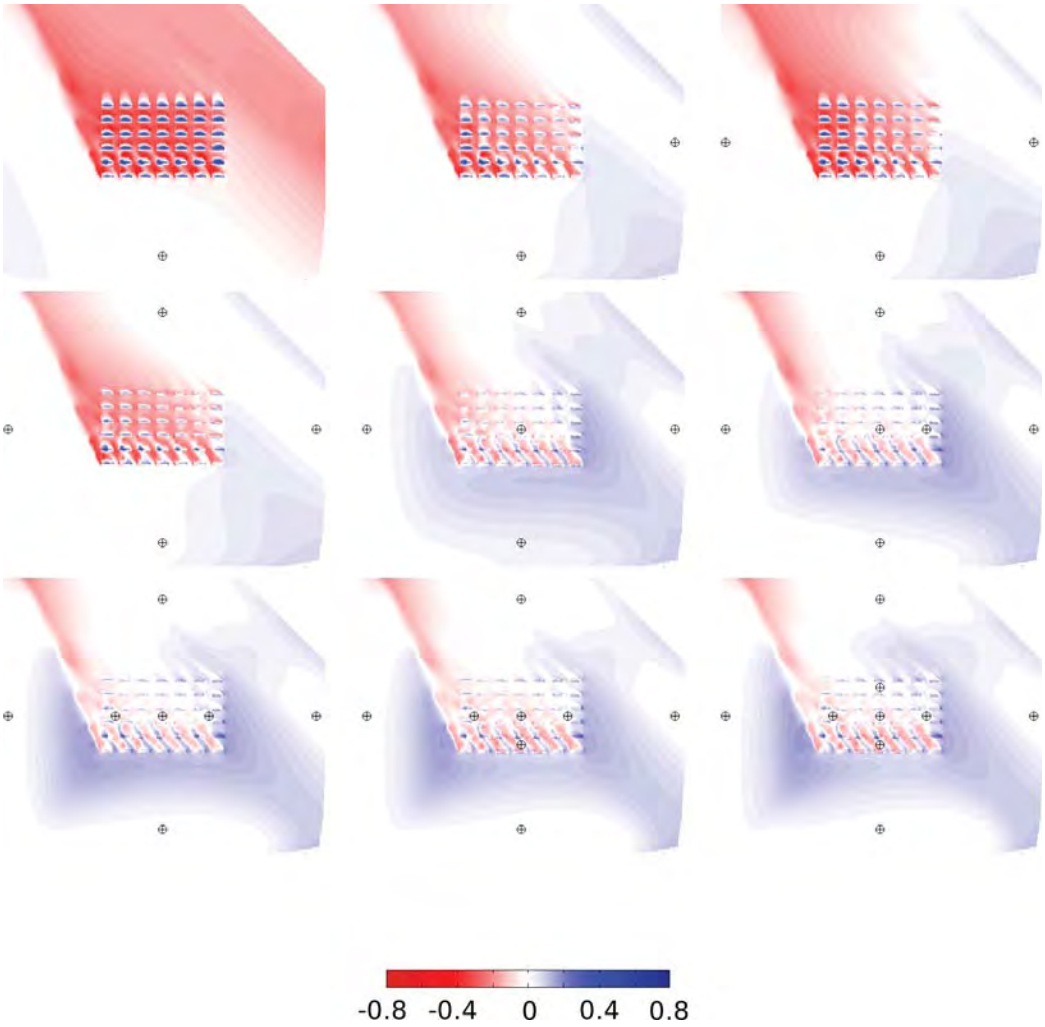

Figure 5.29. Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 2.75 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

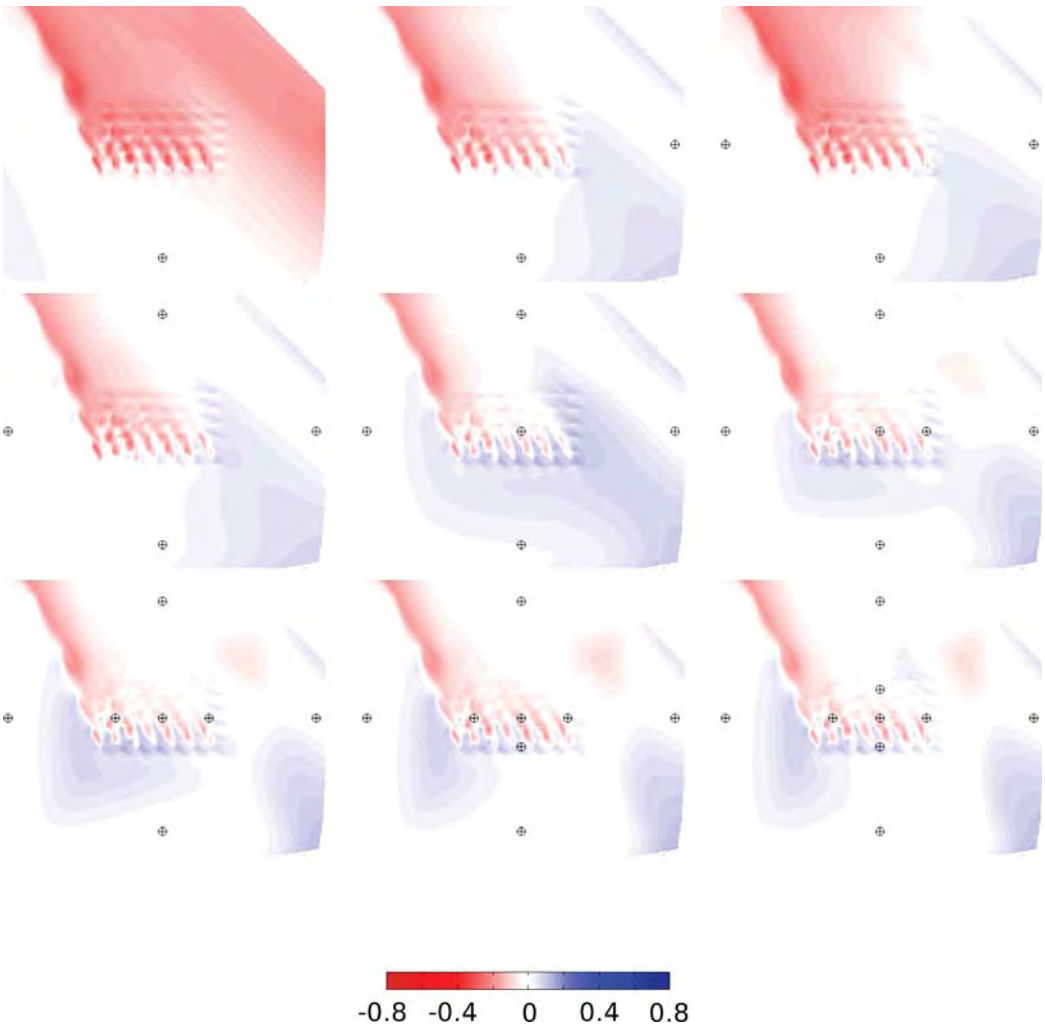

Figure 5.30. Contour slice of the difference in the streamwise (V) velocity of the FLUENT and QUIC-URB solutions at a height of 5.25 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

Table 5.5. Corrected Normalized Root Mean Square Error (NRMSE) of the FLUENT and QUIC-URB solutions for the 6x7 container array and wall at four 1.5 meter vertical sections for the spanwise (U) velocity, streamwise (V) velocity, the vertical (W) velocity and the velocity magnitude (Mag). The FLUENT solution is the estimator and the QUIC-URB solution is the estimated parameter in equation 4.2

Vertical Section	**Velocity Comp.**	**Profiles 1**	**Profiles 1,2**	**Profiles 1-3**	**Profiles 1-4**	**Profiles 1-5**	**Profiles 1-6**	**Profiles 1-7**	**Profiles 1-8**	**Profiles 1-9**
0-1.5 m	U	0.3067	0.0645	0.0612	0.0638	0.0750	0.0699	0.0713	0.0662	0.0630
	V	0.2591	0.0948	0.0923	0.0916	0.1064	0.1009	0.0868	0.0795	0.0780
	W	0.0107	0.0107	0.0107	0.0107	0.0107	0.0107	0.0107	0.0107	0.0107
	Mag	0.5766	0.1700	0.1641	0.1661	0.1921	0.1815	0.1689	0.1564	0.1517
1.5-3.0 m	U	0.4155	0.0294	0.0253	0.0297	0.0426	0.0321	0.0399	0.0388	0.0352
	V	0.3331	0.1204	0.1171	0.1151	0.1334	0.1238	0.1180	0.1172	0.1147
	W	0.0338	0.0338	0.0338	0.0338	0.0338	0.0338	0.0338	0.0338	0.0338
	Mag	0.7824	0.1835	0.1763	0.1785	0.2098	0.1897	0.1916	0.1898	0.1837
3.0-4.5 m	U	0.5001	0.0154	0.0197	0.0143	0.0011	0.0096	0.0048	0.0048	0.0102
	V	0.3636	0.1267	0.1231	0.1200	0.1388	0.1277	0.1305	0.1352	0.1322
	W	0.0828	0.0828	0.0828	0.0828	0.0828	0.0828	0.0828	0.0828	0.0828
	Mag	0.9466	0.2248	0.2257	0.2171	0.2227	0.2201	0.2181	0.2228	0.2252
4.5-6.0 m	U	0.5636	0.0581	0.0626	0.0561	0.0439	0.0498	0.0459	0.0455	0.0520
	V	0.4621	0.1437	0.1393	0.1342	0.1574	0.1455	0.1506	0.1573	0.1526
	W	0.1828	0.1828	0.1828	0.1828	0.1828	0.1828	0.1828	0.1828	0.1828
	Mag	1.2085	0.3844	0.3846	0.3730	0.3840	0.3780	0.3792	0.3855	0.3873

The nine profile solution has the lowest NRMSE for the 0 to 1.5 meter section, while the three profile solution has the lowest for the 1.5 to 3.0 meter section, and the four profile solution has the lowest NRMSE for the 3.0 to 4.5 meter and the 4.5 to 6.0 meter sections.

Figures 5.31 to 5.33 show the QUIC-URB to QUIC-URB comparisons of the multiprofile to single profile cases. The spanwise velocity component shows a significant improvement since the single profile case has very nearly zero spanwise velocity. These figures show the difference in the angle of the wakes of the containers as well. Once again there are some differences in the vertical velocity within the city. However, it is difficult to know which solution is closer to the FLUENT solution since the NRMSE of the vertical velocity component is the same for each of the QUIC-URB cases for each height.

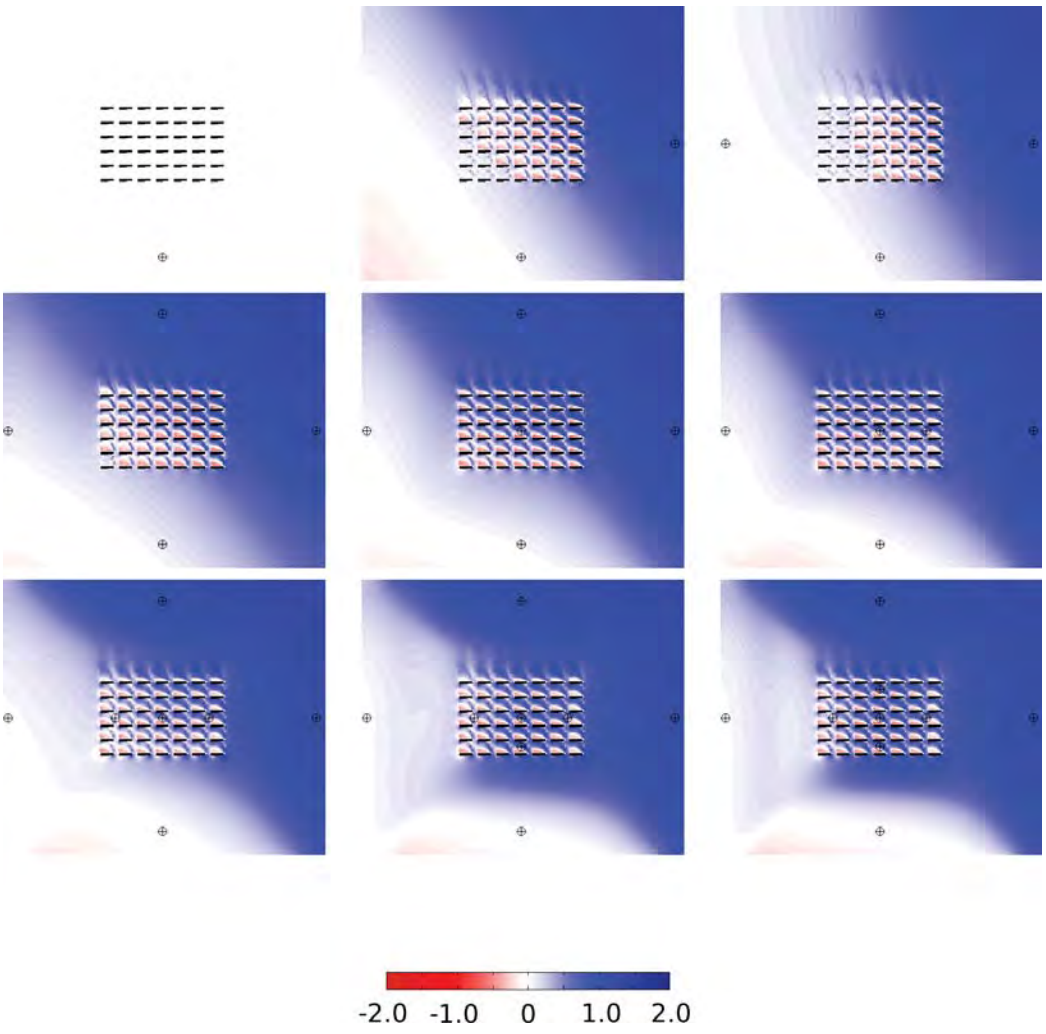

Figure 5.31. Contour slice of the difference in the spanwise (U) velocity of the single sensor QUIC-URB and multisensor QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into multisensor QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

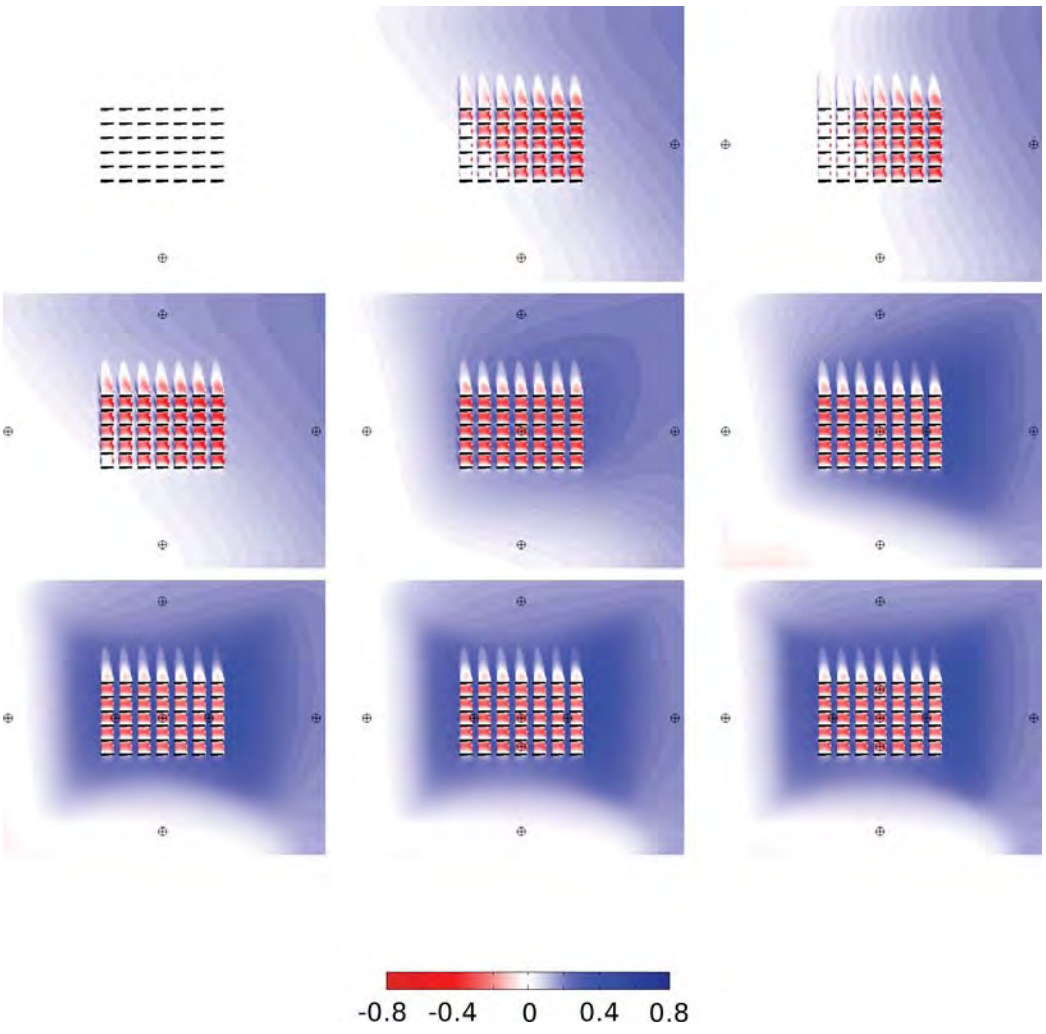

Figure 5.32. Contour slice of the difference in the streamwise (V) velocity of the single sensor QUIC-URB and multisensor QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into multisensor QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

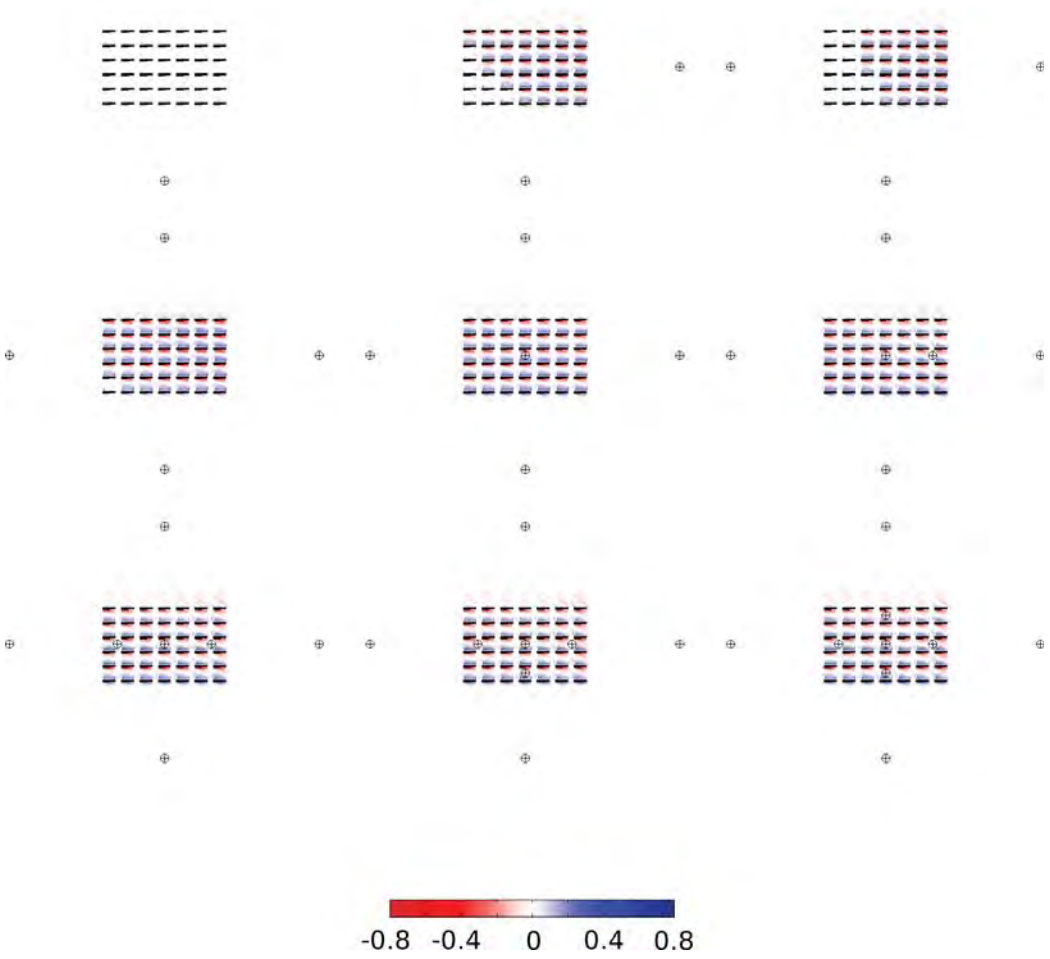

Figure 5.33. Contour slice of the difference in the vertical (W) velocity of the single sensor QUIC-URB and multisensor QUIC-URB solutions at a height of 1.25 meters for the 6x7 array and large wall. Each subplot shows the position of the "sensors" ingested into multisensor QUIC-URB. The upper left subplot is the QUIC-URB solution with a single upwind profile ingested, while the bottom right includes all nine profiles.

CHAPTER 6

CONCLUSION

The analysis presented in this paper showed that the Quasi-3D Barnes Objective Map analysis technique implemented in the QUIC-URB model increased the overall accuracy when predicting the wind field around and through cities that experience spatially inhomogeneous flow. However, the improvement varied depending the number of input profiles, the locations of the profiles, and the magnitude of the variation of the mean windfield. Further research should be done to analyze the sensitivity of final flow field when varying these parameters with this assimilation technique.

The strength of this data assimilation technique is the estimation of non-homogeneous mean windfields. The larger the difference of the inhomogeneities to a horizontally uniform windfield the bigger the improvement that this data assimilation technique has on the mean windfield. In other words, the solution with the large wall topography showed more improvement from the data assimilation technique than the solution with the hemisphere topography and even more improvement than that showed on the baseline array.

Currently this technique applies a uniform weight for every velocity measurement regardless of its location. For sparse input data sets and with ingested velocity measurments that are in an area of localized flow such as the wake of a building, a building street canyon, or in the wake of the city, the local flow velocities get diffused beyond their applicable local region by the guassian weighted averging technique. Unfortunately, this ingestion technique is still in its early stages and does not have automated ways of sorting the profiles or weighting them according to the location inside or outside of these localized flow regions.

These velocity profiles should be limited to affecting a localized area for sparse data sets. Future research could limit the radius of influence of these measurments by applying a lower weight to them in the guassian averaging technique. These weights could be a requirement of the analyst or an automated algorithm based on proximity to buildings. Another method may ignore these local flow measurments during the Guassian technique, then later utilize the magnitude of this data to scale the QUIC-URB empirical parameterizations for that localized region. The major difficulty in this task appears to be the algorithm to automate the selection of these sensors in the local flow regions.

In theory, the accuracy of capturing the mean wind field increases with increased numbers of profiles ingested with this data assimilation technique. At some critical density of sensors it is assumed that the profiles in localized flow regions would not increase the radius of influence of these regions if the rest of the sensors were close enough to limit this radius to an acceptable length. This theory should also be the subject of further research. In practice it is probably not practical to reach this critical density of sensors except in the case of LIDAR data.

Future research should also focus on the effects of varying γ.

Overall, this data assimilation technique can capture the mean wind field of spatially inhomogeneous flows around urban areas which was shown to increase the accuracy of the empirical flow parameterizations and ultimately the overall flowfield. This technique is still in its early stages and needs the analyst to closely monitor the data being ingested to prevent some of its shortcomings from corrupting the mean wind field. It is suggested to follow some of the recommended additional research topics given throughout this paper to better understand the best path to take to automate the ingestion technique further.

APPENDIX A

DATA ASSIMILATION CODE

This section has the data assimilation FORTRAN subroutines implemented in the QUIC-URB model.

A.1 Sensor initialization

This is the main subroutine that implements the Barnes Objective Map analysis scheme for the data assimilation technique in QUIC-URB.

```
subroutine sensorinit
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc
! Subroutine to initialize the initial velocity field (uo,vo,wo) for each time step
!
! This program interpolates irregularly spaced data onto a
! uniform grid using the Barnes Objective Map Analysis
! scheme as implemented in Koch et al., 1983
!
! This program has been modified from Eric Pardyjak's original 2D code to work with
! quicurb 3D
!
! this subroutine uses a quasi-3D barnes objective mapping technique
! quasi-3D is just using sensor height (zref) and sensor wind speed (uref)
! to extrapolate a vertical velocity profile at each sensor location
! to get a velocity at every height at each location
! from these hieght varying velocities, a regular 2D barnes objective map analysis
! is done at each planar height throughout the whole domain.
!
! Called by met_init.f90
!
! Tom Booth 2/17/04
!
! Variable information:
! site_xcoord,site_ycoord,site_zcoord - the coordinates of each site (meters)
! t_site - is the time step for each site
! dir_site - is the direction of the wind for each site at each time step
! vel_site - is the magnitude of the wind for each site at each time step
! total_time_step - is the total number of time steps in a 24 hr period
! num_sites - is the total number of measurement sites
! sgamma - numerical convergence parameter
! lamda - weight parameter (ko)
! deln - average Radius (i.e. computed data spacing)
```

```
! TMB/ERP 9/20/05
! Added a logarithmic interpolation below the the lowest input data point for
! input files that read in wind profile data
!
!ccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccccc

use datamodule
implicit none

integer kk !,site_num
integer iwork, jwork
integer met_input_flag


!real site_time_incr
real site_zref, theta_site, mag_site, umult_site, vmult_site

real yc, xc, rc, rcsum, rcval
real sumw, sumwv, sumwu
real sgamma, lamda, deln
real dxx, dyy, u12, u34, v12, v34
! MAN 04/05/2007 Time varying profile parameters
real bisect
real xtemp,psi_m !erp 9/18/2006 stability variables

real, allocatable :: x(:,:), y(:,:)   !Locations of grid cell centers in meters

allocate(x(nx,ny),y(nx,ny))
!read in whole file on first quasi-time step
if (i_time.eq.1) then
   vk=0.4 !Von Karman's constant
   read(36,*) ! QUIC version header line
   read(36,*)met_input_flag
! MAN 4/17/2007 moved met data read statements to separate subroutines and added
!  architecture for new 3D vectorfield input
   select case(met_input_flag)
      case(0)
         call read_quic_met
      case(1)
         call read_ittmm5_met
      case(2)
         call read_hotmac_met
   endselect
endif !if i_time = 1
!------------------------------------------------------------------------------
!This do loop defines each vertical velocity profile at each sensor site so the
!Barnes Mapping interpolation scheme has a velocity at each site for each height
lp003:   do kk=1,num_sites
if(site_blayer_flag(kk,i_time).eq.4)then  !data entry profile
 allocate(u_data(site_nz_data(kk,i_time)),v_data(site_nz_data(kk,i_time)))
 do ii=1,site_nz_data(kk,i_time)
  ! convert wind speed wind direction into u and v
```

```
  if(site_wd_data(kk,i_time,ii).le.90) then
     u_data(ii)=-site_ws_data(kk,i_time,ii)*  &
       sin((site_wd_data(kk,i_time,ii))*pi/180.)
     v_data(ii)=-site_ws_data(kk,i_time,ii)*  &
       cos((site_wd_data(kk,i_time,ii))*pi/180.)
  endif
  if(site_wd_data(kk,i_time,ii).gt.90.and.site_wd_data(kk,i_time,ii).le.180) then
     u_data(ii)=-site_ws_data(kk,i_time,ii)*  &
       cos((site_wd_data(kk,i_time,ii)-90)*pi/180.)
     v_data(ii)= site_ws_data(kk,i_time,ii)*  &
       sin((site_wd_data(kk,i_time,ii)-90)*pi/180.)
  endif
  if(site_wd_data(kk,i_time,ii).gt.180.and.site_wd_data(kk,i_time,ii).le.270) then
     u_data(ii)= site_ws_data(kk,i_time,ii)*  &
       sin((site_wd_data(kk,i_time,ii)-180)*pi/180.)
     v_data(ii)= site_ws_data(kk,i_time,ii)*  &
       cos((site_wd_data(kk,i_time,ii)-180)*pi/180.)
  endif
  if(site_wd_data(kk,i_time,ii).gt.270.and.site_wd_data(kk,i_time,ii).le.360) then
     u_data(ii)= site_ws_data(kk,i_time,ii)*  &
       cos((site_wd_data(kk,i_time,ii)-270)*pi/180.)
     v_data(ii)=-site_ws_data(kk,i_time,ii)*  &
       sin((site_wd_data(kk,i_time,ii)-270)*pi/180.)
  endif
 enddo
endif
lp002:      do k=2,nz !loop through vertical cell indices
   if(site_blayer_flag(kk,i_time) .lt. 4)then
      theta_site = site_wd_data(kk,i_time,1)
      mag_site = site_ws_data(kk,i_time,1)
      site_zref = site_z_data(kk,i_time,1)
      if(theta_site.le.90) then
         umult_site=-sin((theta_site)*pi/180.)
         vmult_site=-cos((theta_site)*pi/180.)
      endif
      if(theta_site.gt.90.and.theta_site.le.180) then
         umult_site=-cos((theta_site-90)*pi/180.)
         vmult_site=sin((theta_site-90)*pi/180.)
      endif
      if(theta_site.gt.180.and.theta_site.le.270) then
         umult_site=sin((theta_site-180)*pi/180.)
         vmult_site=cos((theta_site-180)*pi/180.)
      endif
      if(theta_site.gt.270.and.theta_site.le.360) then
         umult_site=cos((theta_site-270)*pi/180.)
         vmult_site=-sin((theta_site-270)*pi/180.)
      endif
!power law profile
      if(site_blayer_flag(kk,i_time) .eq. 2)then
         ! MAN 07/25/2008 stretched vertical grid
         u_prof(kk,k)=umult_site*mag_site*((zm(k)/site_zref)**site_pp(kk,i_time))
         v_prof(kk,k)=vmult_site*mag_site*((zm(k)/site_zref)**site_pp(kk,i_time))
```

```
      endif !erp 2/6/2003 end power law profile
!logrithmic velocity profile
    if(site_blayer_flag(kk,i_time) .eq. 1)then
! MAN 05/15/2007 adjust for stability
    if(k.eq.2)then
       if(site_zref*site_rL(kk,i_time) .ge. 0)then
          psi_m=4.7*site_zref*site_rL(kk,i_time)
       else
          xtemp=(1.-15*site_zref*site_rL(kk,i_time))**(0.25)
          psi_m=-2*log(0.5*(1+xtemp))-log(0.5*(1+xtemp**2.))+2*atan(xtemp)-0.5*pi
       endif
       ustar=mag_site*vk/(log(site_zref/site_pp(kk,i_time))+psi_m)
    endif
! end MAN 05/15/2007
       ! MAN 07/25/2008 stretched vertical grid
       if(zm(k)*site_rL(kk,i_time) .ge. 0)then
          psi_m=4.7*zm(k)*site_rL(kk,i_time)
       else
          xtemp=(1.-15*zm(k)*site_rL(kk,i_time))**(0.25)
          psi_m=-2*log(0.5*(1+xtemp))-log(0.5*(1+xtemp**2.))+2*atan(xtemp)-0.5*pi
       endif
       u_prof(kk,k)=(umult_site*ustar/vk)*(log(zm(k)/site_pp(kk,i_time))+psi_m)
       v_prof(kk,k)=(vmult_site*ustar/vk)*(log(zm(k)/site_pp(kk,i_time))+psi_m)
    endif !erp 2/6/2003 end log law profile
!Canopy profile
    if(site_blayer_flag(kk,i_time) .eq. 3)then
       if(k.eq.2)then !only calculate d once
! MAN 05/15/2007 adjust for stability
    if(site_zref*site_rL(kk,i_time) .ge. 0)then
       psi_m=4.7*site_zref*site_rL(kk,i_time)
    else
       xtemp=(1.-15*site_zref*site_rL(kk,i_time))**(0.25)
       psi_m=-2*log(0.5*(1+xtemp))-log(0.5*(1+xtemp**2.))+2*atan(xtemp)-0.5*pi
    endif
    ustar = mag_site*vk/(log(site_zref/site_pp(kk,i_time))+psi_m)
    d = bisect(ustar,site_pp(kk,i_time),site_H(kk,i_time), &
      site_ac(kk,i_time),vk,psi_m)
! end MAN 05/15/2007
     if(site_H(kk,i_time)*site_rL(kk,i_time) .ge. 0)then
        psi_m=4.7*(site_H(kk,i_time)-d)*site_rL(kk,i_time)
     else
        xtemp=(1.-15*(site_H(kk,i_time)-d)*site_rL(kk,i_time))**(0.25)
        psi_m=-2*log(0.5*(1+xtemp))-log(0.5*(1+xtemp**2.))+2*atan(xtemp)-0.5*pi
     endif
     uH = (ustar/vk)*(log((site_H(kk,i_time)-d)/site_pp(kk,i_time))+psi_m);
     if(site_zref .le. site_H(kk,i_time))then
        mag_site=mag_site/(uH*exp(a*(site_zref/site_H(kk,i_time) -1.)))
     else
        if(site_zref*site_rL(kk,i_time) .ge. 0)then
           psi_m=4.7*(site_zref-d)*site_rL(kk,i_time)
        else
           xtemp=(1.-15*(site_zref-d)*site_rL(kk,i_time))**(0.25)
```

```
          psi_m=-2*log(0.5*(1+xtemp))-log(0.5*(1+xtemp**2.))+2*atan(xtemp)-0.5*pi
        endif
        mag_site=mag_site/((ustar/vk)*(log((site_zref-d)/zo)+psi_m))
     endif
     ustar=mag_site*ustar
     uH=mag_site*uH
  endif
  ! MAN 07/25/2008 stretched vertical grid
  if(zm(k) .le. site_H(kk,i_time))then !lower canopy profile
     u_prof(kk,k) = umult_site * uH*exp(site_ac(kk,i_time)&
                    *(zm(k)/site_H(kk,i_time) -1))
     v_prof(kk,k) = vmult_site * uH*exp(site_ac(kk,i_time)&
                    *(zm(k)/site_H(kk,i_time) -1))
  endif
  if(zm(k) .gt. site_H(kk,i_time))then !upper canopy profile
     if(zm(k)*site_rL(kk,i_time) .ge. 0)then
        psi_m=4.7*(zm(k)-d)*site_rL(kk,i_time)
     else
        xtemp=(1.-15*(zm(k)-d)*site_rL(kk,i_time))**(0.25)
        psi_m=-2*log(0.5*(1+xtemp))-log(0.5*(1+xtemp**2.))+2*atan(xtemp)-0.5*pi
     endif
     u_prof(kk,k)=(umult_site*ustar/vk)*&
               (log((zm(k)-d)/site_pp(kk,i_time))+psi_m)

     v_prof(kk,k)=(vmult_site*ustar/vk)*&
               (log((zm(k)-d)/site_pp(kk,i_time))+psi_m)
     endif !end urban canopy TMB 6/16/03
     endif
     endif
!new 2/7/2005 velocity profile entry
     if(site_blayer_flag(kk,i_time) .eq. 4)then  !data entry profile
       !loop through the data points in input profile each time step
        do ii=1,site_nz_data(kk,i_time)
! begin interpolation input velocity to computational grid
! MAN 07/25/2008 stretched vertical grid
    if(zm(k) .eq. site_z_data(kk,i_time,ii))then
       u_prof(kk,k)=u_data(ii)
       v_prof(kk,k)=v_data(ii)
       goto 500
    endif
!erp 9/23/05   logarithmically interpolate to zero velocity at zo below
!              lowest data point
!MAN 01/21/07  logarithmic interpolation uses the first data point instead
!              of the second
   if(zm(k) .gt. 0 .and. zm(k) .lt. site_z_data(kk,i_time,1))then
      if(zm(k) .gt. site_pp(kk,i_time))then
         u_prof(kk,k)= (u_data(1)/(log(site_z_data(kk,i_time,1)/  &
                                site_pp(kk,i_time))))*   &
                                log(zm(k)/site_pp(kk,i_time))
         v_prof(kk,k)= (v_data(1)/(log(site_z_data(kk,i_time,1)/  &
                                site_pp(kk,i_time))))*   &
                                log(zm(k)/site_pp(kk,i_time))
```

```
      else
         u_prof(kk,k)= 0
         v_prof(kk,k)= 0
      endif
      goto 500
   endif
!erp 9/23/05
  if(ii .gt. 1)then
     ! MAN 07/25/2008 stretched vertical grid
     if(zm(k) .gt. site_z_data(kk,i_time,ii-1) .and. &
                    zm(k).lt.site_z_data(kk,i_time,ii))then
       u_prof(kk,k)=((u_data(ii)-u_data(ii-1))/(site_z_data(kk,i_time,ii)- &
                      site_z_data(kk,i_time,ii-1)))   &
                      *(zm(k)-site_z_data(kk,i_time,ii-1)) + u_data(ii-1)
       v_prof(kk,k)=((v_data(ii)-v_data(ii-1))/(site_z_data(kk,i_time,ii)- &
                      site_z_data(kk,i_time,ii-1)))   &
                      *(zm(k)-site_z_data(kk,i_time,ii-1)) + v_data(ii-1)
       goto 500
     endif
  endif
   enddo !end ii loop
 500              continue

   ! extrapolate logarithmically for data beyond input velocity
   ! MAN 07/25/2008 stretched vertical grid
 if(zm(k) .gt. site_z_data(kk,i_time,site_nz_data(kk,i_time))) then
  u_prof(kk,k)=(log(zm(k)/site_z_data(kk,i_time,site_nz_data(kk,i_time)-1))/   &
log(site_z_data(kk,i_time,site_nz_data(kk,i_time))/site_z_data(kk,i_time,   &
   site_nz_data(kk,i_time)-1)))*(u_data(site_nz_data(kk,i_time))-   &
   u_data(site_nz_data(kk,i_time)-1)) + u_data(site_nz_data(kk,i_time)-1)
  v_prof(kk,k)=(log(zm(k)/site_z_data(kk,i_time,site_nz_data(kk,i_time)-1))/   &
   log(site_z_data(kk,i_time,site_nz_data(kk,i_time))/site_z_data(kk,i_time, &
   site_nz_data(kk,i_time)-1)))*(v_data(site_nz_data(kk,i_time))-   &
   v_data(site_nz_data(kk,i_time)-1)) +  &
   v_data(site_nz_data(kk,i_time)-1)
            endif
            if(k .eq. nz) deallocate(u_data,v_data)
         endif !erp 2/6/2003 end data entry
      enddo   lp002       !k=nz
   enddo   lp003       !num_sites kk
! end MAN 04/05/2007
!-------------------------------------------------------------------------------
! find average distance of each measuring station relative to all other
! measuring stations

if(num_sites .eq. 1)then
   do k=2,nz
      uo(:,:,k)=u_prof(1,k)
      vo(:,:,k)=v_prof(1,k)
   enddo
else !Barnes Mapping Scheme
   rcsum=0.
```

```
   do kk=1,num_sites
      rcval=1000000. !ignore anything over 1000 kilometers
      do k=1,num_sites
         xc=site_xcoord(k)-site_xcoord(kk)
         yc=site_ycoord(k)-site_ycoord(kk)
         rc=sqrt(xc**2+yc**2)
         if(rc .lt. rcval .and. k .ne. kk) rcval=rc !shortest distance
      enddo
      rcsum=rcval+rcsum !sum of shortest distances
   enddo
   deln=rcsum/real(num_sites)  !average Radius (i.e. computed data spacing)
   lamda=5.052*(2.*deln/pi)**2 !weight parameter
   !lamda=ffact*lamda !distance dependant factor
   !numerical convergence parameter
   sgamma = 0.2 !gamma=.2 max detail, gamma=1 min detail
! MAN 7/7/2005 var dz conversion
! calculate grid cell center locations in meters
   do j=1,ny
      do i=1,nx
         x(i,j)=real(i-1)*dx-.5*dx !calculating in meters  ddx=(meters/grid)
         y(i,j)=real(j-1)*dy-.5*dy
      enddo
   enddo
! end MAN 7/7/2005

!-------------------------------------------------------------------------------
!first and second barnes pass done for each cell level in z direction

!compute weight of each site on point (i,j)

do j=1,ny
   do i=1,nx
      do kk=1,num_sites
         wm(kk,i,j)=exp(-1/lamda*(site_xcoord(kk)-x(i,j))**2   &
                 -1/lamda*(site_ycoord(kk)-y(i,j))**2)
         wms(kk,i,j)=exp(-1/(sgamma*lamda)*(site_xcoord(kk)-x(i,j))**2   &
                -1/(sgamma*lamda)*(site_ycoord(kk)-y(i,j))**2)
      enddo
      if(sum(wm(:,i,j)) .eq. 0.)then
         wm(:,i,j)=1e-20
      endif
   enddo
enddo
lp004:      do k=2,nz
! interpolate onto the grid
! do first Barnes pass
do j=1,ny
   do i=1,nx
      sumwu=sum(wm(:,i,j)*u_prof(:,k))
      sumwv=sum(wm(:,i,j)*v_prof(:,k))
      sumw=sum(wm(:,i,j))
      uo(i,j,k)=sumwu/sumw
```

```
      vo(i,j,k)=sumwv/sumw
   enddo !i=nx
enddo !j=ny
! before doing the 2nd pass for the Barnes Method
! use a 4-point bilinear interpolation
! scheme to get estimated values at measured point (+)
! using the 1st pass calculated data at grid points (*)
!
!     *       * 1        2
!
!          +      +      !definition of points
!
!     *       *            3        4
!     |    |  |
!     | dxx|  |
!     |       |  !definition of measurements
!     |  ddx  |
!
do kk=1,num_sites
   do j=1,ny
      do i=1,nx
         !find closest grid location on left side of site
         if(x(i,j).lt.site_xcoord(kk)) iwork=i
         !find closest grid location on lower side of site
         if(y(i,j).lt.site_ycoord(kk)) jwork=j
      enddo !i=nx
   enddo !j=ny
   !distance to site point from lower and left sides
   dxx=site_xcoord(kk)-x(iwork,jwork)
   dyy=site_ycoord(kk)-y(iwork,jwork)
! MAN 7/7/2005 fixed interpolation of velocities and var dz conversion
  !upper u interpolated velocity
  u12 = (1-dxx/dx)*uo(iwork,jwork+1,k)+(dxx/dx)*uo(iwork+1,jwork+1,k)
  !lower u interplotaed velocity
  u34 = (1-dxx/dx)*uo(iwork,jwork,k)+(dxx/dx)*uo(iwork+1,jwork,k)
  !total interpolated u velocity
  uoint(kk)=(dyy/dy)*u12+(1-dyy/dy)*u34

  !upper v interpolated velocity
  v12 = (1-dxx/dx)*vo(iwork,jwork+1,k)+(dxx/dx)*vo(iwork+1,jwork+1,k)
  !lower v interplotaed velocity
  v34 = (1-dxx/dx)*vo(iwork,jwork,k)+(dxx/dx)*vo(iwork+1,jwork,k)
  !total interpolated u velocity
  voint(kk)=(dyy/dy)*v12+(1-dyy/dy)*v34
! end MAN 7/7/2005
               enddo !kk=num_sites
! end bilinear interpolation section
! Begin 2nd Barnes pass
do j=1,ny
   do i=1,nx
      sumwu=sum(wms(:,i,j)*(u_prof(:,k)-uoint(:)))
      sumwv=sum(wms(:,i,j)*(v_prof(:,k)-voint(:)))
```

```
      sumw=sum(wms(:,i,j))
      if(sumw.eq.0)then
         uo(i,j,k)=uo(i,j,k)
         vo(i,j,k)=vo(i,j,k)
      else
         uo(i,j,k)=uo(i,j,k) + sumwu/sumw
         vo(i,j,k)=vo(i,j,k) + sumwv/sumw
      endif
   enddo
enddo
! determine the norm between the measurement and interpolation
!      do kk=1,num_sites

!  do j=1,ny
!       do i=1,nx
!
!       if(x(i,j).lt.site_xcoord(kk)) iwork=i
!
!       if(y(i,j).lt.site_ycoord(kk)) jwork=j
!
!       enddo
!       enddo

!      dxx=site_xcoord(kk)-x(iwork,jwork)
!       dyy=site_ycoord(kk)-y(iwork,jwork)

!       u12 = (ddx-dxx)*uo(iwork,jwork+1,k)+dxx*uo(iwork+1,jwork+1,k)
!       u34 = (ddx-ddx)*uo(iwork,jwork,k)+dxx*uo(iwork+1,jwork,k)
!       uoint(kk)=dyy*u12+(1-dyy)*u34

!       v12 = (ddx-dxx)*vo(iwork,jwork+1,k)+dxx*vo(iwork+1,jwork+1,k)
!       v34 = (ddx-dxx)*vo(iwork,jwork,k)+dxx*vo(iwork+1,jwork,k)
!       voint(kk)=dyy*v12+(1-dyy)*v34

            !      unorm=unorm+abs(u_prof(kk,k)-uoint(kk))
!       vnorm=vnorm+abs(v_prof(kk,k)-voint(kk))
!       enddo

            !        unorm=unorm/real(num_sites)
            !        vnorm=vnorm/real(num_sites)
!         write(*,*)
!         write(*,*)'interpolated u-velocity absolute norm: ',unorm
!         write(*,*)'interpolated v-velocity absolute norm: ',vnorm
!         write(*,*)

  enddo   lp004       !k=1:nz
endif

! open(unit=38,file='uofield1.dat',status='unknown')


!if this is the last time loop then deallocate this subroutines variables
```

```
!deallocate arrays
!deallocate(u_prof,v_prof)
deallocate(x,y)
!deallocate(uoint,voint)
!deallocate(wm)

!---------------------------------------------------------------------
!     Writing the uosensorfield.dat file

! if(i_time.eq.1)then

!  open(unit=51,file="uosensorfield.dat",status="unknown")
!      write(51,*)'% Inital sensor velocity field x,y,z,uo,vo,wo'

! endif

!912 format(i9,  '       !Time Increment')
!913 format(i9,  '       !Number of time steps')
!914 format(f9.4,'       !Time')
!916 format(3i6,1x,3(f17.5,1x))
!101   format(6(f11.5,1x))

! if(i_time.eq.1)then
!       write(51,913)num_time_steps
! endif

! write(51,*)'%Start next time step'
! write(51,912)i_time
! write(51,914)time

!   do k=1,nz-1
!        do j=1,ny-1
!        do i=1,nx-1

         !actual coordinate of center of cells
!    x1=(0.5*(real(i+1)+real(i))-1.)*ddx
!         y1=(0.5*(real(j+1)+real(j))-1.)*ddx
!         z1=(0.5*(real(k+1)+real(k))-2.)*ddx

         !        write(51,101)x1,y1,z1,uo(i,j,k),vo(i,j,k),wo(i,j,k)
!      enddo
!       enddo
!       enddo

! if(i_time.eq.num_time_steps) close(51)
!---------------------------------------------------------------------------

!initialize vector fields TMB 7/11/03
u(1:nx,1:ny,1:nz)=0.
v(1:nx,1:ny,1:nz)=0.
w(1:nx,1:ny,1:nz)=0.
! erp initialize upwind array erp 6/18/04
```

```
uo_bl(1:nz)=uo(1,1,1:nz)
vo_bl(1:nz)=vo(1,1,1:nz)

write(44,*)'upstream winds:'
write(44,*)'uo',uo(5,5,10)
write(44,*)'vo',vo(5,5,10)
return
end
```

A.2 Bisection subroutine

This is a subroutine that implements the bisection method for the urban canopy boundary layer profile.

```
function bisect(ustar,zo,H,a,vk,psi_m)result(d)
   !this function uses the bisection method to find the zd displacement height
implicit none
   integer iter
   real zo, H, a, vk, tol,uhc
   real d, fi, d1,d2, fnew, ustar, psi_m !fu,

!this function uses the bisection method to find the root of the specified
!equation

tol=zo/100
fnew=tol*10

!initial two guesses
d1 = zo;
d2 = H;

uhc = (ustar/vk)*(log((H-d1)/zo)+psi_m)
fi = a*uhc*vk/ustar -  H/(H-d1)

! MAN 05/15/2007 fu never used so it is commented out
!         uhc = (ustar/vk)*log((H-d2)/zo)
!         fu = a*uhc*vk/ustar -  H/(H-d2)

iter = 0;

do while(iter.lt.200.and.abs(fnew).gt.tol)
   iter = iter + 1
   d = (d1 + d2) / 2 ! Algorithm for bisect method

   uhc = (ustar/vk)*(log((H-d)/zo)+psi_m)
   fnew = a*uhc*vk/ustar -  H/(H-d)

   if(fnew*fi.gt.0) then
      d1 = d
   elseif(fnew*fi.lt.0)then
      d2 = d
   endif
```

```
enddo

end
```

APPENDIX B

FLUENT INPUTS

This section of the appendix lists the various scripts and input files used to run the fluent cases.

B.1 Fluent case setup

This file was used to setup the fluent case via the text user interface commands. The size of the grid was too large to setup the case through the GUI on a desktop computer since it required more than 24 Gb of RAM. Instead the case was setup on the cluster as the job was submitted using the PBS script in Appendix B.2.

```
;read the mesh
rc
must_cooper.msh

;check the mesh
/mesh/check-verbosity 2
/mesh/check

;define the units that will be used for the input conditions
define/units
area
m2
define/units
density
kg/m3
define/units
length
m
define/units
mass
kg
define/units
pressure
pascal
define/units
temperature
c
define/units
velocity
m/s
```

```
;define the Turbulence model
define/models/viscous/kw-sst
yes
define/models/viscous/kw-low-re-correction
yes

;print out the current zones
define/bc/list-zones

;defining BC types
define/bc/zone-type
downwind_outflow
pressure-outlet
define/bc/zone-type
lside
pressure-outlet
define/bc/zone-type
rside
pressure-outlet
define/bc/zone-type
top
pressure-outlet

;define pressure outlet conditions
define/bc/pressure-outlet
downwind_outflow
no
0
no
no
yes
no
no
yes
1
10
no
no
no

;copy press-out conditions to other bc's
define/bc/copy-bc
downwind_outflow
lside
rside
top

;loading the pre-compiled udf
define/user-define/compiled-functions
load
libudf
```

```
;defining the velocity inlet BC
define/bc/zone-type
upwind_inflow
velocity-inlet

;define the inlet conditions
;using the tke inlet profile
define/bc/velocity-inlet
upwind_inflow
yes
yes
yes
yes
"udf"
"inlet_velocity_profile::libudf"
no
0
yes
no
0
no
1
no
0
yes
yes
yes
"udf"
"tke_inlet_profile::libudf"
no
0.01

;print out the new zones
define/bc/list-zones


;set the reference values
report/ref-val/length
2.5
report/ref-val/velocity
1


;setting the numerical schemes (Solution Methods)
;spatial discretizations
solve/set/discretization-scheme/k
1
solve/set/discretization-scheme/omega
1
solve/set/discretization-scheme/mom
1
```

```
solve/set/discretization-scheme/pressure
12
solve/set/gradient-scheme
yes


;pressure-velocity coupling scheme
solve/set/p-v-coupling
20  ;segregated

q
q
q


;surface monitor creation
;buildings
/solve/monitors/surface/set-monitor
bld
"Integral"
pressure
bld

yes

yes
yes
blds.txt
1


;ground
/solve/monitors/surface/set-monitor
ground
"Integral"
pressure
ground

yes

yes
yes
ground.txt
1

;residual monitor values
solve/monitors/residual/criterion-type
3
solve/monitors/residual/normalize
yes
solve/monitors/residual/n-save
10000
```

```
solve/monitors/residual/scale
yes
yes
yes

;this block prints out the residuals to a file
;================================================================================
(define port)
(set! port (open-output-file "residuals.txt"))

(do ((i 0 (+ i 1)))
         ((= i (length (solver-residuals))))
           (format port "~a ~2t" (car (list-ref (solver-residuals)i))) )
      (newline port)

(define my_run_res
  (lambda ()
       (do ((i 0 (+ i 1)))
     ((= i (length (solver-residuals))))
           (format port "~a ~2t" (cdr (list-ref (solver-residuals)i))) )
      (newline port)
   )

)
(ti-menu-load-string "/solve/execute-commands/add-edit res_out 1 \"iterations\"
      \"(my_run_res())\"")

;================================================================================




;set solution initialization values
solve/init/set-defaults/x-velocity
0
solve/init/set-defaults/y-velocity
1
solve/init/set-defaults/z-velocity
0


;initialize domain and then use fmg init
solve/init/init
solve/init/fmg
yes

;saving case and data file before solving
;wcd
;must_cooper.cas


;iterating
```

```
solve/iter
3000

;saving
wcd
must_cooper-3000.cas.gz

exit
yes
```

B.2 Fluent PBS setup script

This is the script that submits the fluent job to the cluster PBS queueing system. This script makes Fluent read the case setup journal file in Appendix B.1.

```
#!/bin/tcsh
#PBS -S /bin/tcsh
#PBS -N must_wall
#PBS -r n
#PBS -j oe
#PBS -l nodes=10:ppn=12
#PBS -l walltime=3:00:00
##PBS -q debug


setenv OMP_NUM_THREADS 1
limit stacksize 200000 kbytes
setenv KMP_STACKSIZE 100000000
setenv F_UFMTENDIAN big
setenv FORT_BUFFERED 0
setenv MPI_DISPLAY_SETTINGS 1
setenv MPI_IB_RAILS         1
setenv MPI_BUFS_PER_PROC  384
setenv MPI_BUFS_PER_HOST  384

module load ansys/13.0
module unload sunstudio


#------------------------------------------------------------

echo Running on `hostname` in $PBS_O_WORKDIR at : `date`
cd $PBS_O_WORKDIR

set input       = "urban_case_setup.inp"
set output      = "fluent_config.out"
set ncpus       = `cat $PBS_NODEFILE | wc -l`
set fluent_args = "3ddp -t$ncpus -g -i $input -mpi=hp -ssh"
set fluent_args = "$fluent_args -cnf=$PBS_NODEFILE -pinfiniband"

echo "------------------Starting fluent with the following settings:"
echo "Input:   $input"
```

```
echo "NCPU's:  $ncpus"
echo "Output:  $output"
echo "Fluent Args:   $fluent_args"

#this is how to run on l1
time fluent $fluent_args > $output

#archive the residual and monitor files
converge_archive.csh
```

B.3 Fluent PBS SAS script

This is the script that resubmits the fluent job to the cluster PBS queueing system to start the time accurate portion of the simulation.

```
#!/bin/tcsh
#PBS -S /bin/tcsh
#PBS -N MUST_wall
#PBS -r n
#PBS -j oe
#PBS -l nodes=10:ppn=12
#PBS -l walltime=1:30:00
##PBS -q debug


setenv OMP_NUM_THREADS 1
limit stacksize 200000 kbytes
setenv KMP_STACKSIZE 100000000
setenv F_UFMTENDIAN big
setenv FORT_BUFFERED 0
setenv MPI_DISPLAY_SETTINGS 1
setenv MPI_IB_RAILS         1
setenv MPI_BUFS_PER_PROC  384
setenv MPI_BUFS_PER_HOST  384

module load ansys/13.0
module unload sunstudio


#------------------------------------------------------------

echo Running on `hostname` in $PBS_O_WORKDIR at : `date`
cd $PBS_O_WORKDIR

set file_prefix = "must_cooper"
set input       = "fluent.inp"
set startiter   = 3000
set iter        = 1000
set nitnwt      = 8
@ iter_time     = $iter / $nitnwt
@ lastiter      = $startiter + $iter
set case        = "$file_prefix-$startiter.cas.gz"
```

```
set output      = "fluent-$lastiter.out"
set newdata     = "$file_prefix-SAS-$lastiter.dat.gz"
set newcase     = $newdata:s/dat/cas/
set ncpus       = `cat $PBS_NODEFILE | wc -l`
set fluent_args = "3ddp -t$ncpus -g -i $input -mpi=hp -ssh"
set fluent_args = "$fluent_args -cnf=$PBS_NODEFILE -pinfiniband"

#running fluent in batch mode

#writing the fluent input file to the current directory
cat << EOF > $input
rcd "$case"
;--------------writting the residuals out --------------------
(define port)
(set! port (open-output-file "residuals.txt"))

(do ((i 0 (+ i 1)))
         ((= i (length (solver-residuals))))
           (format port "~a ~2t" (car (list-ref (solver-residuals)i))) )
      (newline port)

(define my_run_res
  (lambda ()
       (do ((i 0 (+ i 1)))
     ((= i (length (solver-residuals))))
           (format port "~a ~2t" (cdr (list-ref (solver-residuals)i))) )
      (newline port)
   )

)
(ti-menu-load-string "/solve/execute-commands/add-edit res_out 1 \"iterations\"
      \"(my_run_res())\"")

;-------------------------------------------------------------------------------

;defining the SAS SST turbulence model and standard inputs
/define/models/viscous/sas yes

;define the bounded central differencing momentum scheme if not already set
/solve/set/discretization-scheme/mom 7

;define 2nd order transient
/define/models/unsteady-2nd-order yes

;define dt
/solve/set/time-step 0.0005

;enable autoinit and mod of case to define the max-num-iter-per-timestep
/define/solution-strategy/enable-strategy yes

;set max iter per time step
/solve/max-iterations-per-time-step $nitnwt
```

```
;disable auto init and mod of case to define the max-num-iter-per-timestep
/define/solution-strategy/enable-strategy no

;setting number of timesteps to iterate
/solve/dual-time-iterate $iter_time

wcd "$newdata"
exit
yes
EOF


echo "------------------Starting fluent with the following settings:"
echo "Input:   $input"
echo "Case:    $case"
echo "NCPU's:  $ncpus"
echo "Output:  $output"
echo "# iters: $iter"
echo "Fluent Args:   $fluent_args"
echo "MPI_ROOT:      $MPI_ROOT"

#run on l1
time fluent $fluent_args > $output

converge_archive.csh
```

B.4 Velocity inlet UDF

This User Defined Function (UDF) was written in order to set the proper boundary layer velocity profile and TKE profile at the inlet to the 6x7 container array problems.

```
/************************************************************/
/* UDF that specifies the inlet velocity and tke profiles */
/************************************************************/

#include "udf.h"
/*#include <math.h> */

DEFINE_PROFILE(tke_inlet_profile, thread, index)
{
   real pos[ND_ND];     /* this will hold the position vector */
   real z;              /* this is the height */
   face_t face;         /* this is the boundary face */
   real uref, zref, kmax, zlinemax, zmidpolymax, zupperpolymax;
   real tke;

   uref  = 1.0;    /* meters/sec */
   zref  = 7.29;   /* meters */
   kmax  = 0.035;   /*max k/uref^2 */
   zlinemax  = 0.25*zref;    //z at maximum linear region
```

```
   zmidpolymax = 0.46*zref;  //z at the transition between lower and upper poly
   zupperpolymax = 2.0*zref;  //z at the upper end of the upper poly

   // loop through all the cell faces on the boundary face denoted by 'thread'
   begin_f_loop(face, thread)
      {
        /* this fluent macro returns the x,y,z cell centroid */
        F_CENTROID(pos,face,thread);
        z = pos[2];                      /* x=0, y=1, z=2 */

        // first shot at a tke inlet. tke=0 @ z=0,but dtke/dy != 0 @ z=0, but
        // hopefuly the inlet is far
        // enough upstream that the physics smooth that out. see the MUST_cfd.xls
        // file for details

        if (z<=zlinemax) {
           tke = pow(uref,2) * (kmax/0.25) * (z/zref); /* linear, wall BC*/
        } else if (z>zlinemax && z<=zmidpolymax) {
           tke = pow(uref,2) * (0.2534*pow((z/zref),2) -0.2256*(z/zref) + 0.0752);
        } else if (z>zmidpolymax && z<=zupperpolymax) {
           // 2nd order poly estimate from miskam WTT test
           tke = pow(uref,2)*(0.00092007*pow((z/zref),2)- &
               0.0033688*(z/zref)+0.026047);
        } else {
           // 2nd order poly estimate from miskam WTT test at z/zref =2
           tke = pow(uref,2) * (0.00092007*pow(2,2) - 0.0033688*(2) + 0.026047);
        }

        F_PROFILE(face,thread, index) = tke ;

      }
   end_f_loop(face,thread)

}
/*******************************************************/
/* UDF that specifies the inlet velocity profile      */
/*******************************************************/

DEFINE_PROFILE(inlet_velocity_profile, thread, index)
{
   real pos[ND_ND];     // this will hold the position vector
   real z;              // this is the height
   face_t face;         // this is the boundary face
   real uref, zref, zo; //friction velocity, vonKarmen constant, roughness height
   real umag;


   uref = 1.0;
   zref = 7.29;
   zo   = 0.01;
```

```
   /* loop through all the cell faces on the boundary face denoted by 'thread' */
   begin_f_loop(face, thread)
      {
        F_CENTROID(pos,face,thread); // this returns the x,y,z cell centroid
        z = pos[2];                  // x=0, y=1, z=2
        if (z==0) {
           umag = 0;
        } else {
           umag = uref*log(z/zo)/log(zref/zo);
        }
        F_PROFILE(face,thread, index) = umag;   /*Log Law */

      }
   end_f_loop(face,thread)

}
```

APPENDIX C

QUIC-URB INPUT

This section gives examples of the QUIC-URB input files used for this project.

C.1 QU_simparams.inp

The simulation parameters used for each QUIC-URB case.

```
!QUIC 5.6
320 !nx - Domain Length(X) Grid Cells
272 !ny - Domain Width(Y) Grid Cells
40 !nz - Domain Height(Z) Grid Cells
1 !dx (meters)
1 !dy (meters)
0 !Vertical stretching flag(0=uniform,1=custom,2=parabolic Z,3=parabolic DZ,4=exp)
0.5 !dz (meters)
1 !total time increments
1 !day of the year
0 !UTC conversion
!Time(s) of simulations in decimal hours
0
2 !rooftop flag (0-none, 1-log profile, 2-vortex)
3 !upwind cavity flag (0-none, 1-Rockle, 2-MVP, 3-HMVP)
4 !street canyon flag (0-none,1-Roeckle,2-CPB,3-exp.param.PKK,4-Roeckle w/Fackrel)
1 !street intersection flag (0-off, 1-on)
2 !wake flag (0-none, 1-Rockle, 2-Modified Rockle)
500 !Maximum number of iterations
3 !Residual Reduction (Orders of Magnitude)
0 !Use Diffusion Algorithm (1 = on)
20 !Number of Diffusion iterations
0 !Domain rotation relative to true north (cw = +)
0.0 !UTMX of domain origin (m)
0.0 !UTMY of domain origin (m)
1 !UTM zone
0 !QUIC-CFD Flag
0 !Explosive building damage flag (1 = on)
```

C.2 QU_metparams.inp

This is the format of the file used to define the meteorological conditions within QUIC-URB for the data assimilation technique. This particular file reads

in four files that contain data from a velocity profile extracted from the Fluent solution. An example of the file that is read in is given in Appendix C.7.

```
!QUIC 5.6
0 !Met input flag (0=QUIC,1=ITT MM5,2=HOTMAC)
4 !Number of measuring sites
20 !Maximum size of data points profiles
prof1 !Site Name
!File name
prof1.inp
prof2 !Site Name
!File name
prof2.inp
prof3 !Site Name
!File name
prof3.inp
prof4 !Site Name
!File name
prof4.inp
```

C.3 QU_buildings.inp

This is the building geometry file used to create the 6x7 MUST array with QUIC-URB. It also defines the extent of the boundaries.

```
!QUIC 5.6
0 !x subdomain coordinate (southwest corner) (Cells)
0 !y subdomain coordinate (southwest corner) (Cells)
320 !x subdomain coordinate (northeast corner) (Cells)
272 !y subdomain coordinate (northeast corner) (Cells)
0.1 !Wall roughness length (m)
42 !Number of Structures
!Bld # Group Type Height(m) Width(m) Length(m) Xfo(m) Yfo(m) Zfo(m) Gamma Data
1 1 1 2.5 2 12 100 101 0 0 0
2 2 1 2.5 2 12 118 101 0 0 0
3 3 1 2.5 2 12 136 101 0 0 0
4 4 1 2.5 2 12 154 101 0 0 0
5 5 1 2.5 2 12 172 101 0 0 0
6 6 1 2.5 2 12 190 101 0 0 0
7 7 1 2.5 2 12 208 101 0 0 0
8 8 1 2.5 2 12 100 115 0 0 0
9 9 1 2.5 2 12 118 115 0 0 0
10 10 1 2.5 2 12 136 115 0 0 0
11 11 1 2.5 2 12 154 115 0 0 0
12 12 1 2.5 2 12 172 115 0 0 0
13 13 1 2.5 2 12 190 115 0 0 0
14 14 1 2.5 2 12 208 115 0 0 0
15 15 1 2.5 2 12 100 129 0 0 0
16 16 1 2.5 2 12 118 129 0 0 0
17 17 1 2.5 2 12 136 129 0 0 0
18 18 1 2.5 2 12 154 129 0 0 0
19 19 1 2.5 2 12 172 129 0 0 0
```

```
20 20 1 2.5 2 12 190 129 0 0 0
21 21 1 2.5 2 12 208 129 0 0 0
22 22 1 2.5 2 12 100 143 0 0 0
23 23 1 2.5 2 12 118 143 0 0 0
24 24 1 2.5 2 12 136 143 0 0 0
25 25 1 2.5 2 12 154 143 0 0 0
26 26 1 2.5 2 12 172 143 0 0 0
27 27 1 2.5 2 12 190 143 0 0 0
28 28 1 2.5 2 12 208 143 0 0 0
29 29 1 2.5 2 12 100 157 0 0 0
30 30 1 2.5 2 12 118 157 0 0 0
31 31 1 2.5 2 12 136 157 0 0 0
32 32 1 2.5 2 12 154 157 0 0 0
33 33 1 2.5 2 12 172 157 0 0 0
34 34 1 2.5 2 12 190 157 0 0 0
35 35 1 2.5 2 12 208 157 0 0 0
36 36 1 2.5 2 12 100 171 0 0 0
37 37 1 2.5 2 12 118 171 0 0 0
38 38 1 2.5 2 12 136 171 0 0 0
39 39 1 2.5 2 12 154 171 0 0 0
40 40 1 2.5 2 12 172 171 0 0 0
41 41 1 2.5 2 12 190 171 0 0 0
42 42 1 2.5 2 12 208 171 0 0 0
```

C.4 QU_fileoptions.inp

The file used for the QU_fileoptions.inp file.

```
!QUIC 5.6
3   !output data file format flag (1=ascii, 2=binary, 3=both)
0   !flag to write non-mass conserved initial field (uofield.dat)
0   !flag to write the file uosensorfield.dat, the initial sensor velocity field
0   !flag to write the file QU_staggered_velocity.bin used by QUIC-Pressure
```

C.5 template.proj

The template used for the 'project.proj' files for each QUIC-URB run.

```
Creator:
     Tom Booth
Date:
     30-June-2011
Notes:
     None
Inner Grid Inlet Profile:
     Data Point Entry
Roof Top (Inner Grid):
     Vortex
Wind Angle(Inner Grid):
     180
Velocity Ref(Inner Grid):
     0
```

```
# of Buildings(Inner Grid):
     42
Inner Grid Scale:
     dx = 1 dy = 1 dz = 0.5
Nested Grid Flag:
     0
Inner Grid Location,X:
     0
Inner Grid Location,Y:
     0
~
```

C.6 template.info

The file that was used as the 'project.info' file for QUIC-URB.

```
BUILDING INFO
height !Color scheme
colormap !Color Map
off !Transparent
on !Numbered
on !Bld Reshape Symbols
off !Bld Auto-Correlation
0.6 !Constant Color (RGB)
0.6
0.6
GROUND INFO
0.8 !Color (RGB)
0.8
0.8
off !Transparent
on !Map
off !Topolines
on !Topoline labels
20 !Topo Levels
on !Axes
2 !Axes Update Speed
```

C.7 prof9.inp

An example of the format used for the QUIC-URB data assimilation technique. The QU_metparams.inp file C.2 is the input file in QUIC-URB that points to this file when using the data assimilation technique.

```
prof9 !Site name
160.5 !X coordinate
164.5 !Y coordinate
0.000000 !Decimal time (military time i.e. 0130 = 1.5)
4 !site boundary layer flag (1=log,2=exp,3=urban canopy,4=discrete data points)
0.100000 !site zo
20 !enter number of wind speed data points
```

```
!Height (m),Speed (m/s), Direction (deg relative to true N)
0.000100  0.000000  311.953963
0.250000   0.156014   311.953963
0.750000   0.063070   275.003126
1.250000   0.059275   202.345746
1.750000   0.134645   172.953606
2.250000   0.277821   169.219956
2.750000   0.464298   171.931398
3.250000   0.603398   174.020340
3.750000   0.695611   175.117682
4.250000   0.765482   175.727446
4.750000   0.824651   176.119629
5.250000   0.877946   176.468405
5.750000   0.923569   176.866777
6.250000   0.961409   177.344863
6.750000   0.991849   177.879141
7.250000   1.016061   178.417399
7.750000   1.035208   178.904875
8.250000   1.050661   179.295717
8.750000   1.063606   179.573133
9.250000   1.074836   179.750586
```

APPENDIX D

POSTPROCESSING SCRIPTS

This section contains all the Matlab, Perl, csh, and Tecplot scripts used to process the FLUENT and QUIC-URB data files.

D.1 Tecplot velocity vector field extraction macro

This was the Tecplot360 macro used to read in the Fluent case and data files and write out the nodecentered columnized data file.

```
#!MC 1200
# Created by Tecplot 360 build 12.2.0.9077
$!VarSet |MFBD| = '/lustre/work/tbooth/projects/thesis/must_semisphere_cooper'
$!READDATASET  '"STANDARDSYNTAX" "1.0" "LoadOption" "CaseAndData" \
   "FILENAME_CaseFile" \
   "must_semisphere_cooper-SAS-10000.cas.gz" "FILENAME_DataFile" \
   "must_semisphere_cooper-SAS-10000.dat.gz"\
   "GridZones" "CellsAndBoundaries" "IncludeParticleData" "No" "AllPolyZones" \
   "No" "AverageToNodes" "Yes"\
   "AveragingMethod" "Arithmetic" "SaveUncompressedFiles" "No"'
  DATASETREADER = 'Fluent Data Loader'
$!WRITEDATASET  "|MFBD|/must_semisphere_cooper_nodecenter.dat"
  INCLUDETEXT = NO
  INCLUDEGEOM = NO
  INCLUDECUSTOMLABELS = NO
  INCLUDEAUTOGENFACENEIGHBORS = YES
  ASSOCIATELAYOUTWITHDATAFILE = NO
  VARPOSITIONLIST =  [1-3,14,20,26]
  BINARY = NO
  USEPOINTFORMAT = YES
  PRECISION = 9
  TECPLOTVERSIONTOWRITE = TECPLOTCURRENT
$!RemoveVar |MFBD|
```

D.2 Tecplot profile extraction macro

This is the Tecplot360 macro used to extract the vertical velocity profiles from the Fluent solutions to import into QUIC-URB.

```
#!MC 1200
# Created by Tecplot 360 build 12.2.0.9077
$!READDATASET  '"STANDARDSYNTAX" "1.0" "LoadOption" "CaseAndData" \
```

```
   "FILENAME_CaseFile" \
   "must_semisphere_cooper-SAS-10000.cas.gz" "FILENAME_DataFile" \
   "must_semisphere_cooper-SAS-10000.dat.gz"
   "GridZones" "SelectedZones" "AllPolyZones" "No" "AverageToNodes" "Yes" \
   "AveragingMethod" "Arithmetic" \
   "SaveUncompressedFiles" "No" "IncludeParticleData" "Yes" "VarNameList" \
   "Pressure"+"X Velocity"+"Y Velocity"+"Z Velocity"+"Turbulent Kinetic Energy"'
  DATASETREADER = 'Fluent Data Loader'

$!EXTRACTFROMPOLYLINE
  EXTRACTTHROUGHVOLUME = YES
  EXTRACTLINEPOINTSONLY = NO
  INCLUDEDISTANCEVAR = NO
  NUMPTS = 19
  EXTRACTTOFILE = YES
  FNAME = 'prof4.dat'
  RAWDATA
2
0.5 450.5 0.25
0.5 450.5 9.25

$!EXTRACTFROMPOLYLINE
  EXTRACTTHROUGHVOLUME = YES
  EXTRACTLINEPOINTSONLY = NO
  INCLUDEDISTANCEVAR = NO
  NUMPTS = 19
  EXTRACTTOFILE = YES
  FNAME = 'prof2.dat'
  RAWDATA
2
150.5 336.5 0.25
150.5 336.5 9.25

$!EXTRACTFROMPOLYLINE
  EXTRACTTHROUGHVOLUME = YES
  EXTRACTLINEPOINTSONLY = NO
  INCLUDEDISTANCEVAR = NO
  NUMPTS = 19
  EXTRACTTOFILE = YES
  FNAME = 'prof1.dat'
  RAWDATA
2
0.5 225.5 0.25
0.5 225.5 9.25

$!EXTRACTFROMPOLYLINE
  EXTRACTTHROUGHVOLUME = YES
  EXTRACTLINEPOINTSONLY = NO
  INCLUDEDISTANCEVAR = NO
  NUMPTS = 19
  EXTRACTTOFILE = YES
  FNAME = 'prof3.dat'
```

```
  RAWDATA
2
-151.5 336.5 0.25
-151.5 336.5 9.25

$!EXTRACTFROMPOLYLINE
  EXTRACTTHROUGHVOLUME = YES
  EXTRACTLINEPOINTSONLY = NO
  INCLUDEDISTANCEVAR = NO
  NUMPTS = 19
  EXTRACTTOFILE = YES
  FNAME = 'prof9.dat'
  RAWDATA
2
0.5 364.5 0.25
0.5 364.5 9.25

$!EXTRACTFROMPOLYLINE
  EXTRACTTHROUGHVOLUME = YES
  EXTRACTLINEPOINTSONLY = NO
  INCLUDEDISTANCEVAR = NO
  NUMPTS = 19
  EXTRACTTOFILE = YES
  FNAME = 'prof6.dat'
  RAWDATA
2
45.5 336.5 0.25
45.5 336.5 9.25

$!EXTRACTFROMPOLYLINE
  EXTRACTTHROUGHVOLUME = YES
  EXTRACTLINEPOINTSONLY = NO
  INCLUDEDISTANCEVAR = NO
  NUMPTS = 19
  EXTRACTTOFILE = YES
  FNAME = 'prof8.dat'
  RAWDATA
2
0.5 308.5 0.25
0.5 308.5 9.25

$!EXTRACTFROMPOLYLINE
  EXTRACTTHROUGHVOLUME = YES
  EXTRACTLINEPOINTSONLY = NO
  INCLUDEDISTANCEVAR = NO
  NUMPTS = 19
  EXTRACTTOFILE = YES
  FNAME = 'prof7.dat'
  RAWDATA
2
-46.5 336.5 0.25
-46.5 336.5 9.25
```

```
$!EXTRACTFROMPOLYLINE
  EXTRACTTHROUGHVOLUME = YES
  EXTRACTLINEPOINTSONLY = NO
  INCLUDEDISTANCEVAR = NO
  NUMPTS = 19
  EXTRACTTOFILE = YES
  FNAME = 'prof5.dat'
  RAWDATA
2
0.5 336.5 0.25
0.5 336.5 9.25
```

D.3 Perl format conversion tool (tec2txt)

This was the Perl script that converted the extracted velocity profiles from tecplot format to a simple columnized txt format.

```
#! /usr/bin/perl

use strict;
use File::Copy;
use Cwd;


my $purpose =
"#This script converts an ascii tecplot file into an ascii data file
that is space delimited with the header info at the top of the columns
\n";

#required inputs for this script
my @inputs = qw/FILEIN FILEOUT/;
my $script = "tec2txt";

my $narg    = scalar(@ARGV);
my $ninputs = scalar(@inputs);

die "\nUsage: $script  @inputs\n\n$purpose\n" unless ($narg == $ninputs);

my $filein    = $ARGV[0];
my $fileout   = $ARGV[1];

open IN, "<$filein" or die "I died trying to open the file '$filein'!\n";
open OUT, ">$fileout" or die "I died trying to open the file '$fileout'!\n";

my $varflag = 0;
print OUT "#";
while (<IN>) {
   #skip lines after ZONE
   last if (/ZONE/) ;
   $_ = (split(/"/,$_))[1];
```

```
   s/ /_/g;
   print OUT "$_   ";
}

print OUT  "\n";
while (<IN>) {
   unless (/DT=\(/) {
      print OUT "$_";
   }
}

close OUT;
close IN;
system "columnize -y $fileout $fileout";
print "done!\n";
```

D.4 Perl format conversion tool (datastripquic)

This was the Perl script that stripped out the unneeded variables from the FLUENT data text file

```
#! /usr/bin/perl -w

use strict;

die "Usage:     datastrip_dir_calc filein fileout
     This function keeps the variables in the file that match the variable names
     given to it in the input and it calculates the magnitude and angle of the
     wind at each height using U and V velocities.
     The output is formatted for Quic-Urb files.
     \n" unless (scalar(@ARGV) == 2);

open IN, "<$ARGV[0]" or die "couldn't open $ARGV[0]\n";
open OUT, ">$ARGV[1]" or die "couldn't open $ARGV[1]\n";

my @vars = qw/Height  Speed  Direction/;
my $pi   = 3.1415926;

#create a indexed value for each column name
my %nam2i ;
my @lines = <IN>;
my $lastline = -1;

foreach (@lines) {
   last unless (/#/);
   $lastline++;
}

$_ = $lines[$lastline];
s/#//;
my @namelist = split;
my $i = 0;
```

```
foreach my $name (@namelist) {
   $nam2i{$name} = $i;
   $i++;
}

#print the right format for QUIC-URB
my $sensorname = (split(/\./,$ARGV[0]))[0];
$_ = $lines[$lastline + 1];
my @row1 = split;
my $x = $row1[$nam2i{X}];
my $y = $row1[$nam2i{Y}];
my $deg = sprintf('%.6f',(180 + atan2( $row1[$nam2i{X_Velocity}] ,
   $row1[$nam2i{Y_Velocity}] ) *180/$pi)) ;


print OUT "$sensorname !Site name\n";
print OUT "$x !X coordinate\n";
print OUT "$y !Y coordinate\n";
print OUT "0.000000 !Decimal time (military time i.e. 0130 = 1.5)\n";
print OUT "4 !site boundary layer flag (1 = log, 2 = exp, 3 = urban canopy,
   4 = discrete data points)\n";
print OUT "0.100000 !site zo\n";
print OUT "20 !enter number of wind speed data points\n";
print OUT "!Height (m),Speed (m/s), Direction (deg relative to true N)\n";
print OUT "0.000100  0.000000  $deg\n";

print "stripping out the un-needed variables from $ARGV[0]....\n";
my $value;
foreach (@lines) {
   unless (/#/) {
      chomp;
      my @row = split;
      foreach my $var (@vars) {
         $_ = $var;
         if (/Height/) {
            $value = sprintf('%.6f',$row[$nam2i{Z}]);
         } elsif (/Direction/) {
            #the fluent files have flow in the Y direction (180)
            #180 is North and 270 is east
            $value = sprintf('%.6f',(180 + atan2( $row[$nam2i{X_Velocity}] ,
               $row[$nam2i{Y_Velocity}] ) *180/$pi)) ;
         } elsif (/Speed/) {
            $value = sprintf('%.6f',(sqrt($row[$nam2i{X_Velocity}]**2 +
               $row[$nam2i{Y_Velocity}]**2)));
         }
         print OUT "$value   ";
      }
      print OUT "\n";
   }
}
```

```
close IN;
close OUT;
```

D.5 Tecplot nodecenter conversion

This csh script strips off the header of the tecplot file and removes the face connectivity information.

Where the $nphead variable is the number of header lines and the $nvert variable is the number of vertices.

```
#! /bin/csh

set infile = must_semisphere_cooper_nodecenter_tec.dat
set outfile = must_semisphere_cooper_nodecenter.txt
set nvert = 10921642
set nhead = 24

@ nvertphead = $nvert + $nhead

echo "X Y Z U V W" >  $outfile
head -$nvertphead $infile | tail -$nvert >> $outfile
```

D.6 Fluent file reduction function

This Matlab function reduces the amount of data in the Fluent data file output from Tecplot.

```
function Fluent_file_reduction(wind)

%this file reduces the raw FLUENT wind files to the size used in QUICURB
[filename,pathname] = uigetfile('*.*','Open fluent wind file');
if filename == 0
    cancel = 1;
else
    cancel = 0;
    fname1 = fullfile(pathname,filename);
    %get current directory
    pdir = cd;
    cd(pathname);   %change to directory of file
end

%starts in same directory of previous file
[filename,pathname] = uiputfile('*.dat*','Save reduced fluent data wind file');

%change back to original directory
cd(pdir)

if filename == 0
    cancel = 1;
else
    cancel = 0;
```

```
    fname2 = fullfile(pathname,filename);
end

if ~cancel
    fid1 = fopen(fname1);
    fid2 = fopen(fname2,'wt+');

    fprintf(fid2,'x \t y \t z \t u \t v \t w \n');
    ncols = 6;

    %read the header
    output = fscanf(fid1,'%s %s %s %s %s %s\n',[1,ncols]);
    endread = 0;
    while endread == 0

        %x y z u v w
        try
            output = fscanf(fid1,'%f %f %f %f %f %f\n',[1,ncols]);
            x = output(1);
            y = output(2);
            z = output(3);
            u = output(4);
            v = output(5);
            w = output(6);
         catch
            endread = 1
         end

        if ~endread
            %keeping all data within the quic domain
            %plus adding space beyond this domain so the interpolation
            %function has values to interpolate to the boundaries of the QUIC
            %domain of the same size
            if (x>=-160) & (x<=160) & (y>=200) & (y<=472) & (z>=0) & (z<=10)
                x = x + 160;
                y = y - 200;
                fsize = fprintf(fid2,'%4.5f\t %4.5f\t %4.5f\t %4.5f\t %4.5f\t
                  %4.5f\n',[x,y,z,u,v,w]);
            end


        end

    end

    fclose(fid1);
    fclose(fid2);
end
```

D.7 Fluent file sort function

This Matlab function sorts the data in the Fluent file to match the ordered format of the QUIC-URB files.

```
function Fluent_file_sort(wind)

%this file converts the unordered FLUENT wind files to the same format as
%the QUIC wind files

[filename,pathname] = uigetfile('*.*','Open reduced fluent wind file');
if filename == 0
    cancel = 1;
else
    cancel = 0;
    fname1 = fullfile(pathname,filename);
    pdir = cd;
    cd(pathname);   %change to directory of file
end
%starts in same directory of previous file
[filename,pathname] = uiputfile('*.dat*','Save sorted fluent data wind file');

%change back to original directory
cd(pdir)

if filename == 0
    cancel = 1;
else
    cancel = 0;
    fname2 = fullfile(pathname,filename);
end

if ~cancel
    fid1 = fopen(fname1);
    fid2 = fopen(fname2,'wt+');

    headerlines = fscanf(fid1,'%1c %1c %1c %1c %1c %1c\n',[1,6]);

    wind = fscanf(fid1,'%f %f %f %f %f %f\n',[6,inf]);
    fclose(fid1);

    windsorted = sortrows(wind',[3, 2, 1]);
    clear wind
    fprintf(fid2,'%4.5f\t %4.5f\t %4.5f\t %4.5f\t %4.5f\t %4.5f\n', \\
      windsorted(:,1:6)');
    fclose(fid2);
end
done = 1
```

D.8 Unique data perl script

This Perl script deletes all of the duplicate points in the Fluent data file.

```
#! /usr/bin/perl -w

use strict;
my $purpose = qq~
removes duplicate points from a file with a list of points

\n
~;

my @inputs = qw/file_in file_out/;
my @codes   = qw(columnize); # Required codes placed in here

# Argument Parsing
my $filein     = $ARGV[0];
my $fileout    = $ARGV[1];

# Print usage if necessary
die "\nUsage: unique_data.pl @inputs\n$purpose"
   if (scalar(@ARGV)!=scalar(@inputs));

# Check for dependant codes
&code_checker(@codes) if (scalar(@codes) > 0);
#-----------------------------------------------------------------------------#
#                                    MAIN                                     #
#-----------------------------------------------------------------------------#
open IN, "<$filein" or die "didn't open $filein dummy!\n";
open OUT,">$fileout" or die "didn't open $fileout dummy!\n";

my @fin = <IN>;
my %seen = ();
my @uniq = ();
print "looking for duplicate points.....\n";
my $dup = 0;
foreach (@fin) {
   #substitute white space for a single _ to get rid of possible
   #dubplicate points with different spacing
   chomp;
   s/[\s]+/_/g;
   unless ($seen{$_}) {
      # if we get here we have not seen it before
      $seen{$_} = 1;
      s/_/  /g;
      push @uniq, $_;
      print OUT "$_ \n";
   } else {
      $dup = $dup + 1;
   }
}

my $pword = "points";
$pword = "point" if ($dup == 1);
print "\n\nFound $dup duplicate $pword in the file $filein\n";
```

```
close IN;
close OUT;
system "columnize -y $fileout $fileout";
#-------------------------------------------------------------------------------#
#                                 SUBROUTINES                                   #
#-------------------------------------------------------------------------------#
sub code_checker {
   use strict;
   my @codes = @_;
   # Checks for your ability to run this
   foreach my $code (@codes ) {
      my $status = system "which $code >> /dev/null";
      if ($status != 0) {
         die "   Error - Could not find '$code' in your path";
      }
   }
}
```

D.9 Matlab sphere function

This Matlab function ('zero_sphere.m') fills the empty sphere in the Fluent data file with zero's.

```
function zero_sphere

%this function creates a matrix of zero's for the must-hemisphere case
%Since matlab keeps crashing when interpolating from fluent to quic-urb.

%center of sphere in fluent coordinates
xc = 110;
yc = 440;
zc = -30;

%converting to Quic-urb coordinates
xc = xc + 160;
yc = yc - 200;

rad = 70;
r   = 34.5;

n = 50;
np1 = n+1;

[x,y,z] = sphere(n);

is = 1;
ie = np1;

for i=30:1:70
   X(is:ie,1:np1) = x*i+xc;
   Y(is:ie,1:np1) = y*i+yc;
```

```
   Z(is:ie,1:np1) = z*i+zc;

   is = is + np1;
   ie = ie + np1;
end
fid = fopen('zero_sphere.txt','w');
imax = is-1;
k=0;
%turning the imax-x-np1 arrays into vectors
for i=1:imax
  for j=1:np1
    k=k+1;
    xv(k) = X(i,j);
    yv(k) = Y(i,j);
    zv(k) = Z(i,j);
    if (zv(k) >= 0) && (zv(k) <= 10)
      fprintf(fid,'%4.5f\t %4.5f\t %4.5f\t %4.5f\t %4.5f\t %4.5f\n',
         xv(k),yv(k),zv(k),0,0,0);
    end
  end
end

fclose(fid);
```

D.10 Matlab Z-limit file reduction function

This Matlab function ('Fluent_file_reduction_zlimit.m') reduces the size of the Fluent data file by slicing it up into 2 meter sections along the z-axis. It is very similar to the other file reduction function, however it only reduces the file along the z-axis and doesn't need to convert the fluent coordinates into QUIC-URB coordinates.

```
function Fluent_file_reduction(wind)

%this file reduces the FLUENT wind files to the size used in the QUICURB project

[filename,pathname] = uigetfile('*.*','Open fluent wind file');
if filename == 0
    cancel = 1;
else
    cancel = 0;
    fname1 = fullfile(pathname,filename);
    %get current directory
    pdir = cd;
    cd(pathname);   %change to directory of file
end

%starts in same directory of previous file
[filename,pathname] = uiputfile('*.dat*','Save reduced fluent data wind file');
```

```
%change back to original directory
cd(pdir)

if filename == 0
    cancel = 1;
else
    cancel = 0;
    fname2 = fullfile(pathname,filename);
end

if ~cancel
    [pathname2, name, ext, ver] = fileparts(fname2);
    fname02 = fullfile(pathname2,[name,'0-2','.dat']);
    fname24 = fullfile(pathname2,[name,'2-4','.dat']);
    fname46 = fullfile(pathname2,[name,'4-6','.dat']);
    fname68 = fullfile(pathname2,[name,'6-8','.dat']);
    fname810 = fullfile(pathname2,[name,'8-10','.dat']);

    fid1 = fopen(fname1);
    fid02 = fopen(fname02,'wt+');
    fid24 = fopen(fname24,'wt+');
    fid46 = fopen(fname46,'wt+');
    fid68 = fopen(fname68,'wt+');
    fid810 = fopen(fname810,'wt+');

    %fprintf(fid2,'x \t y \t z \t u \t v \t w \n');
    ncols = 6;

    %read the header
    %output = fscanf(fid1,'%s %s %s %s %s %s\n',[1,ncols]);
    endread = 0;
    while endread == 0

        %x y z u v w
        try
            output = fscanf(fid1,'%f %f %f %f %f %f\n',[1,ncols]);
            x = output(1);
            y = output(2);
            z = output(3);
            u = output(4);
            v = output(5);
            w = output(6);
         catch
            endread = 1
         end

        if ~endread
            %keeping all data within the quic domain
            %plus adding space beyond this domain so the interpolation
            %function has values to interpolate to the boundaries of the QUIC
            %domain of the same size
            if z<=2
```

```
                fsize = fprintf(fid02,'%4.5f\t %4.5f\t %4.5f\t %4.5f\t %4.5f\t
                  %4.5f\n',[x,y,z,u,v,w]);
            elseif z > 2 && z <=4
                fsize = fprintf(fid24,'%4.5f\t %4.5f\t %4.5f\t %4.5f\t %4.5f\t
                  %4.5f\n',[x,y,z,u,v,w]);
            elseif z >4 && z <=6
                fsize = fprintf(fid46,'%4.5f\t %4.5f\t %4.5f\t %4.5f\t %4.5f\t
                  %4.5f\n',[x,y,z,u,v,w]);
            elseif z>6 && z<=8
                fsize = fprintf(fid68,'%4.5f\t %4.5f\t %4.5f\t %4.5f\t %4.5f\t
                  %4.5f\n',[x,y,z,u,v,w]);
            elseif z>8
                fsize = fprintf(fid810,'%4.5f\t %4.5f\t %4.5f\t %4.5f\t %4.5f\t
                  %4.5f\n',[x,y,z,u,v,w]);
            end


        end

    end

    fclose(fid1);
    fclose(fid02);
    fclose(fid24);
    fclose(fid46);
    fclose(fid68);
    fclose(fid810);
    done = 1
end
```

D.11 Fluent interpolation function

This Matlab function interpolates the Fluent data to the QUIC-URB grid.

```
function fluent_interp(varargin)

if nargin == 0
    cancel = 0;
    [filename1,pathname1] = uigetfile('*.*','Open QUIC wind file');
    if filename1 == 0
        cancel = 1;
    else
        fname1 = fullfile(pathname1,filename1);
    end

    [filename2,pathname2] = uigetfile('*.*','Open fluent wind file');
    if filename2 == 0
        cancel = 1;
    else
        fname2 = fullfile(pathname2,filename2);
    end
```

```
     [filename3,pathname3] = uiputfile('*.mat*','Save interpolated data wind file');
    if filename3 == 0
        cancel = 1;
    else
        fname3 = fullfile(pathname3,filename3);
    end
else
    fname1 = varargin{1};
    fname2 = varargin{2};
    fname3 = varargin{3};
    cancel = 0;
end
if cancel == 0
    %reading the fluent file
    wind = load(fname2);
    wind2(:,1:4) = wind(:,1:4); %clearing room in RAM
    clear wind
    %wind1 = read_windfield(fname1);
    %simply load in the QUIC-URB format if it was already saved as a *.mat file
    %in the meshgrid format
    load(fname1);
    clear wind1.u wind1.v wind1.w

    gu = griddata3(wind2(:,1),wind2(:,2),wind2(:,3),wind2(:,4),wind1.x{1}(:,:,:), &&
                   wind1.y{1}(:,:,:),wind1.z{1}(:,:,:));
    clear wind2
    save(fname3,'gu');
    clear gu

    %reading the fluent file
    wind = load(fname2);
    wind2(:,1:3) = wind(:,1:3); %clearing room in RAM
    wind2(:,4) = wind(:,5);
    clear wind

    gv = griddata3(wind2(:,1),wind2(:,2),wind2(:,3),wind2(:,4),wind1.x{1}(:,:,:), &&
                   wind1.y{1}(:,:,:),wind1.z{1}(:,:,:));
    clear wind2
    save(fname3,'gv','-append');

    clear gv
    %reading the fluent file
    wind = load(fname2);
    wind2(:,1:3) = wind(:,1:3); %clearing room in RAM
    wind2(:,4) = wind(:,6);
    clear wind

    gw = griddata3(wind2(:,1),wind2(:,2),wind2(:,3),wind2(:,4),wind1.x{1}(:,:,:), &&
                   wind1.y{1}(:,:,:),wind1.z{1}(:,:,:));
    clear wind2
    save(fname3,'gw','-append');
```

```
    %wind1 = read_windfield(fname1);
    load(fname1);
    save(fname3,'wind1','-append');

end
```

D.12 QUIC-URB file reader

This Matlab function reads the QUIC-URB data format.

```
function outpt = read_windfield(windname,varargin)
%this function reads the windfield associated with the multi-directional
%Quic-Urb code.  It has a different format than earlier versions
%This function also converts the windfield data into meshgrid format

if nargin > 1
    waitbar_handle = varargin{1};
else
    waitbar_handle = waitbar(0,...
        ['Please wait...Loading Wind Data ']);
end
filename = fopen(windname, 'r');
if (filename <= 0)
    %if file cannot be opened abort function and display error message
    h = errordlg([windname,' does not exist'],...
        'Unable to Locate File');
    uiwait(h)
else
    %file opens with header lines that are not repeated at each section
    %get rid of this line here
    discard = fgets(filename);
    total_incr = fscanf(filename, '%f%*[^\n]', 1);
    discard = fgets(filename);

    for i = 1:total_incr
        %the header line
        discard = fgets(filename);
        incr = fscanf(filename, '%f%*[^\n]', 1);
        discard = fgets(filename);
        current_time = fscanf(filename, '%f%*[^\n]', 1);
        discard = fgets(filename);
        %generating a waitbar so the user can see the progress
        waitbar(i/total_incr,waitbar_handle,...
            ['Please wait...Loading Wind Data (t=',num2str(current_time),')'])

        wind = fscanf(filename, '%f %f %f %f %f %f\n', [6,inf]);

        %these are the number of cells
        x_index = length(unique(wind(1,:)));
        y_index = length(unique(wind(2,:)));
        z_index = length(unique(wind(3,:)));
```

```
        %the starting of the grid cells in meters
        x_min = wind(1,1);
        y_min = wind(2,1);
        z_min = wind(3,1);

        %these are the actual dimensions in meters
        x_max = max(unique(wind(1,:)));
        y_max = max(unique(wind(2,:)));
        z_max = max(unique(wind(3,:)));

        %this the the length of each cell
        dx = (x_max - x_min)/(x_index-1);
        dy = (y_max - y_min)/(y_index-1);
        dz = (z_max - z_min)/(z_index-1);

        %setting the wind field into meshgrid format
        m=1;
        for k = 1:z_index
            for j = 1:y_index
                windfield.u{i}(j,1:x_index,k)=wind(4,m:m+x_index-1);
                windfield.v{i}(j,1:x_index,k)=wind(5,m:m+x_index-1);
                windfield.w{i}(j,1:x_index,k)=wind(6,m:m+x_index-1);
                m=m+x_index;
            end
        end

        clear wind

        %generating a meshgrid type position matrix in meters
        [windfield.x{i}, windfield.y{i}, windfield.z{i}] = ...
            meshgrid(x_min:dx:x_max,y_min:dy:y_max,z_min:dz:z_max);
    end
end

fclose(filename);
outpt = windfield;
delete(waitbar_handle)
```

D.13 Batch interpolation function

This Matlab batch function interpolates the vertical Fluent solution sections to the QUIC-URB mesh.

```
function batch_interp

%must cooper
fluent_interp must_6x7_velocity.mat &&
   must_cooper_nodecenter_reduced_sorted_unique_zlimit1bh.dat &&
   must_cooper_interp1bh.mat
fluent_interp must_6x7_velocity.mat &&
   must_cooper_nodecenter_reduced_sorted_unique_zlimit2bh.dat &&
```

```
   must_cooper_interp2bh.mat
fluent_interp must_6x7_velocity.mat &&
   must_cooper_nodecenter_reduced_sorted_unique_zlimit3bh.dat &&
   must_cooper_interp3bh.mat
fluent_interp must_6x7_velocity.mat &&
   must_cooper_nodecenter_reduced_sorted_unique_zlimit4bh.dat &&
   must_cooper_interp4bh.mat

%must hemisphere hybrid
fluent_interp must_6x7_velocity.mat  &&
   must_semisphere_hybrid_nodecenter_reduced_sorted_zlimit1bh.dat &&
   must_semisphere_hyb_interp1bh.mat
fluent_interp must_6x7_velocity.mat  &&
   must_semisphere_hybrid_nodecenter_reduced_sorted_zlimit2bh.dat &&
   must_semisphere_hyb_interp2bh.mat
fluent_interp must_6x7_velocity.mat  &&
   must_semisphere_hybrid_nodecenter_reduced_sorted_zlimit3bh.dat &&
   must_semisphere_hyb_interp3bh.mat
fluent_interp must_6x7_velocity.mat  &&
   must_semisphere_hybrid_nodecenter_reduced_sorted_zlimit4bh.dat &&
   must_semisphere_hyb_interp4bh.mat

%must wall
fluent_interp must_6x7_velocity.mat &&
   must_wall_cooper_nodecenter_reduced_sorted_unique_zlimit1bh.dat &&
   must_wall_cooper_interp1bh.mat
fluent_interp must_6x7_velocity.mat &&
   must_wall_cooper_nodecenter_reduced_sorted_unique_zlimit2bh.dat &&
   must_wall_cooper_interp2bh.mat
fluent_interp must_6x7_velocity.mat &&
   must_wall_cooper_nodecenter_reduced_sorted_unique_zlimit3bh.dat &&
   must_wall_cooper_interp3bh.mat
%this one crashes matlab
%fluent_interp must_6x7_velocity.mat &&
%   must_wall_cooper_nodecenter_reduced_sorted_unique_zlimit4bh.dat &&
%   must_wall_cooper_interp4bh.mat
```

D.14 Matlab solution comparison function

This Matlab function takes the difference between the Fluent velocity field and the QUIC-URB velocity field. It also calculates the NRMSE and creates error contour plots for the U, V, and W velocities at incremental z-slices through the velocity difference field.

```
function compare_flow_fields(varargin)
%this function compares two flow fields (uoutmat.dat) and returns the
%difference between them (err) for every point of the flow field (i.e.
%size(err) = size(uoutmat.dat)

errtype = 'true';
```

```
if nargin > 3
    sensornum = varargin{4};
    sensorx   = varargin{5};
    sensory   = varargin{6};
    fid       = varargin{7};
else
    sensornum = 1;
    sensorx   = 0;
    sensory   = 0;
end

if isequal(errtype,'true')
    errtitle = ' (Fluent - QUIC-URB) ';
    if sensornum > 1
        errtitle = [errtitle,num2str(sensornum),' sensor'];
    else
        errtitle = [errtitle,num2str(sensornum),' sensors'];
    end
else
    errtitle = ' (True % Relative Error) {1 sensor}';
end

if nargin == 0
    [filename1,pathname1] = uigetfile('*.mat','Open interpolated fluent file');
    fname1 = fullfile(pathname1,filename1);
    if (filename1 == 0)
        cancel = 1;
    else
        [filename2,pathname2] = uigetfile({'*.dat','*.mat'},'Open Quic wind file');
        fname2 = fullfile(pathname2,filename2);
        if (filename2 == 0)
            cancel = 1;
        else
            cancel = 0;
        end
    end
else
    fname1 = varargin{1};
    fname2 = varargin{2};
    cancel = 0;
end

[pathname2, name, ext, ver] = fileparts(fname2);

if ~cancel
    load(fname1);
    if (ext == '.dat')
        wind1 = read_windfield(fname2);
        save(fullfile(pathname2,[name,'.mat']),'wind1');
    else
        load(fname2);
    end
```

```
[xsize,ysize,zsize] = size(gu);
uref =  4;   %m/s
if isequal(errtype,'true')
    err.u = gu - wind1.u{1};
    err.v = gv - wind1.v{1};
    err.w = gw - wind1.w{1};
else
    err.u = abs((gu - wind1.u{1})/uref)*100;
    err.v = abs((gv - wind1.v{1})/uref)*100;
    err.w = abs((gw - wind1.w{1})/uref)*100;
end
err.x = wind1.x{1};
err.y = wind1.y{1};
err.z = wind1.z{1};
%clear gu gv gw wind1

plottype = 'z';
component = ['u','v','w'];
%current z values
val       = -0.25:0.5:9.75;
%val = [-0.25,0.25];

if nargin > 2
    fname3 = varargin{3};
else
    [filename3,pathname3] = uigetfile('*.inp','Open QUIC building file');
    if filename3 == 0
        cancel = 1;
    else
        cancel = 0;
        fname1 = fullfile(pathname3,filename3);
    end
end

if ~cancel
    outpt = read_QU_build(fname3);
    %skip the first index, since it is all zeros
    for j=1:3
        for i=2:length(val)
            plot_diff_contour(err,outpt.bld,plottype,component(j),val(i),i, &
                 errtype,errtitle,sensornum,sensorx,sensory);
        end
    end
end

%turn this to 1 to calculate the nrmse of the fields
nrmse_flag = 1;

if nrmse_flag
    num_of_points = 0;
    sum_sq_diff   = 0;
    u_sum_sq_diff = 0;
```

```
v_sum_sq_diff = 0;
w_sum_sq_diff = 0;

fluent_min =  999;
fluent_max = -999;

[ny, nx, nz] = size(gu);
for i=1:nx
    for j=1:ny
        for k=1:nz
            %lots of NaN's in the Fluent file since it had a symmetry
            %BC
            if  ~isnan(gu(j,i,k))
                num_of_points =+ 1;
                usum_sq_diff =+ (gu(j,i,k) - wind1.u{1}(j,i,k))^2;
                vsum_sq_diff =+ (gv(j,i,k) - wind1.v{1}(j,i,k))^2;
                wsum_sq_diff =+ (gw(j,i,k) - wind1.w{1}(j,i,k))^2;

                quic_mag = sqrt(wind1.u{1}(j,i,k)^2 + wind1.v{1}(j,i,k)^2 &
                            + wind1.w{1}(j,i,k)^2);
                fluent_mag = sqrt(gu(j,i,k)^2 + gv(j,i,k)^2 +gw(j,i,k)^2);

                if fluent_mag < fluent_min
                    fluent_min = fluent_mag;
                elseif fluent_mag > fluent_max
                    fluent_max = fluent_mag;
                end

                sum_sq_diff =+ (quic_mag - fluent_mag)^2;
            end
        end
    end
end

%min and max
umin = min(min(min(gu)));
umax = max(max(max(gu)));
vmin = min(min(min(gv)));
vmax = max(max(max(gv)));
wmin = min(min(min(gw)));
wmax = max(max(max(gw)));

%root mean square of the error
rmse_u = sqrt(usum_sq_diff/num_of_points);
rmse_v = sqrt(vsum_sq_diff/num_of_points);
rmse_w = sqrt(wsum_sq_diff/num_of_points);
rmse   = sqrt(sum_sq_diff/num_of_points);
%normalized root mean square error
nrmse_u = rmse_u/(umax-umin);
nrmse_v = rmse_v/(vmax-vmin);
nrmse_w = rmse_w/(wmax-wmin);
%    rms_uvw = sqrt((nrmse_u^2 + nrmse_v^2 + nrmse_w^2)/3)
```

```
        nrmse_sum = nrmse_u+nrmse_v+nrmse_w;

        fprintf(fid, '%s    %6.4f  %6.4f  %6.4f  %6.4f\n', fname2, ...
            nrmse_u, nrmse_v, nrmse_w, nrmse_sum);
    end
end
```

D.15 Matlab batch comparison function

This Matlab function runs the 'compare_flow_fields.m' function for each set of QUIC-URB solutions for each given Fluent solution. It also saves the contour plots and the NRMSE data.

```
function batch_compare(varargin)

build_file = 'QU_buildings.inp';

fluent = 'must_cooper_interp.mat';
vel_files = {
'quicfiles/must_cooper_p1_QU_velocity.dat',
'quicfiles/must_cooper_p2_QU_velocity.dat',
'quicfiles/must_cooper_p3_QU_velocity.dat',
'quicfiles/must_cooper_p4_QU_velocity.dat',
'quicfiles/must_cooper_p5_QU_velocity.dat',
'quicfiles/must_cooper_p6_QU_velocity.dat',
'quicfiles/must_cooper_p7_QU_velocity.dat',
'quicfiles/must_cooper_p8_QU_velocity.dat',
'quicfiles/must_cooper_p9_QU_velocity.dat'};

%fluent = 'must_semisphere_cooper_interp.mat';
%vel_files = {
%'quicfiles/must_semisphere_p1_QU_velocity.dat',
%'quicfiles/must_semisphere_p2_QU_velocity.dat',
%'quicfiles/must_semisphere_p3_QU_velocity.dat',
%'quicfiles/must_semisphere_p4_QU_velocity.dat',
%'quicfiles/must_semisphere_p5_QU_velocity.dat',
%'quicfiles/must_semisphere_p6_QU_velocity.dat',
%'quicfiles/must_semisphere_p7_QU_velocity.dat',
%'quicfiles/must_semisphere_p8_QU_velocity.dat',
%'quicfiles/must_semisphere_p9_QU_velocity.dat'};

%fluent = 'must_wall_cooper_interp.mat';
%vel_files = {
%'quicfiles/must_wall_p1_QU_velocity.dat',
%'quicfiles/must_wall_p2_QU_velocity.dat',
%'quicfiles/must_wall_p3_QU_velocity.dat',
%'quicfiles/must_wall_p4_QU_velocity.dat',
%'quicfiles/must_wall_p5_QU_velocity.dat',
%'quicfiles/must_wall_p6_QU_velocity.dat',
%'quicfiles/must_wall_p7_QU_velocity.dat',
%'quicfiles/must_wall_p8_QU_velocity.dat',
%'quicfiles/must_wall_p9_QU_velocity.dat'};
```

```
a = 1;

%cell centered positions of all the profiles to be extracted
x{1} = [160.5];
y{1} = [25.5];
x{2} = [310.5, 160.5];
y{2} = [136.5,  25.5];
x{3} = [  9.5, 310.5, 160.5];
y{3} = [136.5, 136.5,  25.5];
x{4} = [160.5,   9.5, 310.5, 160.5];
y{4} = [250.5, 136.5, 136.5,  25.5];
x{5} = [160.5, 160.5,   9.5, 310.5, 160.5];
y{5} = [136.5, 250.5, 136.5, 136.5,  25.5];
x{6} = [205.5, 160.5, 160.5,   9.5, 310.5, 160.5];
y{6} = [136.5, 136.5, 250.5, 136.5, 136.5,  25.5];
x{7} = [114.5, 205.5, 160.5, 160.5,   9.5, 310.5, 160.5];
y{7} = [136.5, 136.5, 136.5, 250.5, 136.5, 136.5,  25.5];
x{8} = [160.5, 114.5, 205.5, 160.5, 160.5,   9.5, 310.5, 160.5];
y{8} = [108.5, 136.5, 136.5, 136.5, 250.5, 136.5, 136.5,  25.5];
x{9} = [160.5, 160.5, 114.5, 205.5, 160.5, 160.5,   9.5, 310.5, 160.5];
y{9} = [164.5, 108.5, 136.5, 136.5, 136.5, 250.5, 136.5, 136.5,  25.5];


fid = fopen('NRMSE.txt', 'w+');
fprintf(fid, '#NRMSE of each component\n');
fprintf(fid, ['#velocity_file               ',...
    'u        v        w        sum\n']);

for i=1:length(vel_files)

  d = vel_files{i}(1:length(vel_files{i})-4);

  if exist(d)
      rmdir(d,'s')
  end
  mkdir(d)
  sensory = y{i};
  sensorx = x{i};
  %must_cooper
  sensornum = vel_files{i}(24);
  %%must_semisphere_cooper
  %sensornum = vel_files{i}(28);
  %%must_wall_cooper
  %sensornum = vel_files{i}(22);

  compare_flow_fields(fluent,vel_files{i},build_file,sensornum,sensorx,sensory,fid)
  movefile('*.png',d);
end
fclose(fid);
end
```

REFERENCES

[1] B. Leitl, K. Bezpalcova, and F. Harms, *Wind Tunnel Modeling of the MUST Experiment*. PhD thesis, Meteorological Institute, University of Hamburg, Hamburg, Germany; Institute of Thermomechanics, Academy of Sciences of the Czech Republic, Prague, Czech Republic, 2007.

[2] M. Balczó and J. Eichhorn, "Refined MISKAM simulations of the Mock Urban Setting Test," *Croatian Meteorological Journal*, 2009.

[3] George Mason University, *Annual atmospheric transport and dispersion modeling conference*, (Fairfax, VA), 1997-2010.

[4] K. J. Allwine, J. H. Shinn, G. E. Streit, K. L. Clawson, and M. J. Brown, "Overview of URBAN 2000: A multiscale field study of dispersion through an urban environment," *Bull. Amer. Meteor. Soc., 83, 521–536*, 2002.

[5] K. J. Allwine, M. J. Leach, L. W. Stockham, J. S. Shinn, R. P. Hosker, J. F. Bowers, and J. C. Pace, "Overview of Joint Urban 2003: An atmospheric dispersion study in Oklahoma city," *Amer. Meteor. Soc., J7.1*, 2004.

[6] M. Hussain and B. E. Lee, "A wind tunnel study of the mean pressure forces acting on large groups of low rise buildings.," *J Wind Eng Ind Aerodyn*, vol. 8, pp. 207–225, 1980.

[7] E. Yee and C. A. Biltoft, "Concentration fluctuation measurements in a plume dispersing through a regular array of obstacles.," *Defence R&D Canada - Suffield, P.O. Box 4000, Medicine Hat, Alberta, TIA 8k6 Canada*, 2003.

[8] C. A. Biltoft, "Customer report for Mock Urban Setting Test," tech. rep., West Desert Test Center, U.S. Army Dugway Proving Ground, Dugway, Utah, 2001.

[9] J. L. McElroy, "A comparative study of urban and rural dispersion.," *J Appl Meteorol.*, vol. 8, pp. 19–31, 1969.

[10] S. R. Hanna, J. White, Y. Zhou, and A. Kosheleva, "Analysis of ju2003 and msg05 meteorological and tracer data.," *In: 6th AMS symposium urban environment, Atlanta, GA*, 2006.

[11] F. E. Camelli, W. J. Coirier, O. R. Hansen, A. Huber, S. Kim, S. Hanna, and M. J. Brown, "An intercomparison of four computational fluid dynamics models: transport and dispersion around madison square garden," *American Meteorological Society*, 2006.

[12] A. A. Gowardhan, E. R. Pardyjak, I. Senocak, and M. J. Brown, "A CFD-based wind solver for an urban fast response transport and dispersion model," *Environ Fluid Mech*, vol. 11, pp. 439–464, 2010.

[13] W. J. Coirier and S. Kim, "Summary of CFD urban results in support of the Madison square garden and urban dispersion program field tests.," *American Meteorological Society*, 2006.

[14] W. J. Coirier, S. Kim, F. Chen, and M. Tuwari, "Evaluation of urban scale contamination transport and dispersion modeling using loosely coupled CFD and mesoscale models.," *American Meteorological Society*, 2006.

[15] J. Eichhorn and M. Balczó, "Flow and dispersal simulations of the Mock Urban Setting Test," *Croatian Meteorological Journal*, vol. 43, pp. 67–72, 2008. -Proceedings of the 12th International Conference on Harmonization within Atmospheric Dispersion Modelling for Regulatory Purposes (HARMO12).

[16] F. E. Camelli, R. Löhner, and S. R. Hanna, "VLES study of MUST experiment," *43rd AIAA Aerospace Sciences Meeting and Exhibit*, 2005.

[17] M. Neophytou, A. A. Gowardhan, and M. J. Brown, "An inter-comparison of three urban wind models using oklahoma city joint urban 2003 wind field measurements.," *Journal of Wind Engineering and Industrial Aerodynamics*, vol. 99, no. DOI 10.1016/j.physletb.2003.10.071, pp. 357–368, 2011.

[18] E. R. Pardyjak, B. Singh, A. Norgren, and P. Willemsen, "Using video gaming technology to achieve low-cost speed up of emergency response urban dispersion simulations.," *American Meteorological Society*, 2007.

[19] B. Singh, E. R. Pardyjak, A. Norgren, and P. Willemsen, "Accelerating urban fast response Lagrangian dispersion simulations using inexpensive graphics processor parallelism," *Environmental Modeling and Software*, vol. 26, pp. 739–750, 2011.

[20] R. Röckle, *Bestimmung der Stromungsverhaltnisse im Bereich komplexer Bebauungsstrukturen.* PhD thesis, der Technischen Hochschule Darmstadt, Germany, 1990. Ph.D. dissertation.

[21] E. R. Pardyjak and M. J. Brown, "Evaluation of a fast-response urban wind model: Comparison to single building wind-tunnel data.," *Proceedings of the 3rd International Symposium on Environmental Hydraulics.*, 2001.

[22] E. R. Pardyjak and M. J. Brown, "Fast response modeling of two building street canyon.," *4th AMS Urb. Env. Symp., Norfolk, VA.*, 2002.

[23] E. R. Pardyjak, M. J. Brown, and N. Bagal, "Improved velocity deficit parameterizations for a fast response urban wind model.," *Symp. Planning, Nowcasting, and Forcasting Urb. Zone, 84th Annual Mtg. AMS, Seattle, WA.*, 2004.

[24] N. Bagal, E. R. Pardyjak, B. Singh, and M. J. Brown, "Implementation of rooftop recirculation parameterization in the QUIC fast response urban wind model.," *Am Meteorol Soc, CDROM 6.10*, 2004.

[25] N. Bagal, E. R. Pardyjak, and M. J. Brown, "Improved upwind cavity parameterization for a fast response urban wind model.," *Am Meteorol Soc, CDROM P1.13*, 2004.

[26] N. Bagal, "Development and testing of empirical parameterizations for the quick urban and industrial complex model.," Master's thesis, University of Utah, 2005.

[27] B. Singh, E. R. Pardyjak, M. J. Brown, and M. D. Williams, "Testing of a far-wake parameterization for a fast response urban wind model.," *6th AMS Urb. Env. Symp., Atlanta, GA.*, 2006.

[28] S. U. Pol, N. Bagal, B. Singh, M. J. Brown, and E. R. Pardyjak, "Implementation of a new rooftop recirculation parameterization into the QUIC fast response urban wind model.," *6th AMS Urb. Env. Symp., Atlanta, GA.*, 2006.

[29] B. Singh, B. S. Hansen, M. J. Brown, and E. R. Pardyjak, "Evaluation of the QUIC-URB fast response urban wind model for a cubical building array and wide building street canyon," *Environ Fluid Mech*, vol. 8, no. DOI 10.1007/s10652-008-9084-5, pp. 281–312, 2008.

[30] J. Clark and P. Klein, "Implementation of a traffic-produced turbulence scheme into the fast-response model QUIC.," *6th AMS Urb. Env. Symp., Atlanta, GA.*, 2006.

[31] M. D. Williams, M. J. Brown, and E. R. Pardyjak, "Development and testing of a dispersion model for flow around buildings.," *4th AMS Symp. Urban Env., Norfolk, VA.*, 2002.

[32] T. M. Booth and E. R. Pardyjak, "Validation of a data assimilation technique for an urban wind model.," *6th AMS Urb. Env. Symposium. Atlanta GA.*, 2006.

[33] R. B. Stull, *An Introduction to Boundary Layer Meteorology*, vol. 1. Kluwer Academic Publishers, 1st ed., 1988.

[34] T. R. Oke, "The energetic basis for the urban heat island.," *Quart. J. R. Met. Soc*, vol. 108, pp. 1–24, 1982.

[35] S. E. Koch, M. DesJardins, and P. Kocin, "An interactive barnes objective map analysis scheme for use with satellite and conventional data.," *Journal of Climate and Applied Meteorology*, vol. 22, pp. 1487–1502, 1983.

[36] S. L. Barnes, "Mesoscale objective analysis using weighted time-series observations.," *NOAA Tech. Memo. ERL NSSL-92, National Severe Storms laboratory, Norman OK 73069, 60 pp [NTIS COM-73-10781].*, 1973.

[37] R. M. Cionco, "A mathematical model for air flow in a vegetative canopy," *Journal of Applied Meteorology*, vol. 4, no. 4, pp. 517–522, 1965.

[38] R. M. Cionco, "Application of the ideal canopy flow concept to natural and artificial roughness elements," *R&D Technical Report ECOM-5372*, vol. 4, no. 4, 1971.

[39] R. M. Cionco, "Analysis of canopy index values for various canopy densities," *Journal of Applied Meteorology*, 1978.

[40] R. W. McDonald, "Modelling the mean velocity profile in the urban canopy layer.," *Boundary-Layer Meteorology*, vol. 4, pp. 517–522, 2000.

[41] F. R. Menter, "Two-equation eddy-viscosity turbulence models for engineering applications.," *AIAA Journal*, vol. 32, no. 8, 1994.

[42] D. C. Wilcox, *Turbulence Modeling for CFD*. DCW Industries Inc., 2nd ed., 1998.

[43] F. R. Menter, M. Kuntz, and R. Bender, "A scale adaptive simulation model for turbulent flow predictions," *AIAA*, 2003.

[44] A. Inc., *ANSYS FLUENT Theory Guide*. ANSYS Inc., 2010.